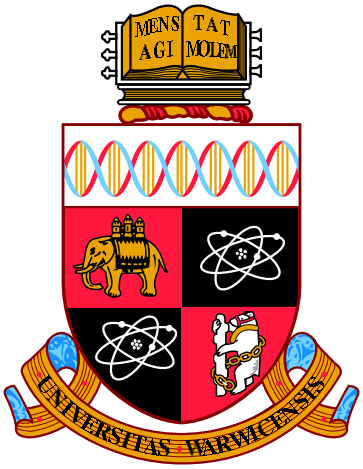


Bayesian Models for Scalable Machine Learning

by

Valerio Perrone

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Statistics

October 2018

Contents

Acknowledgments **v**

Declarations **vi**

Abstract **vii**

1 Introduction **1**

1.1 Bayesian nonparametrics 2

1.1.1 Exchangeability 3

1.1.2 Completely random measures 4

1.2 Deep learning 5

1.3 Inference 6

1.4 Outline and contributions of thesis 9

2 Bayesian Nonparametric Dynamic Feature Models **11**

2.1 Introduction 12

2.2 Background 14

2.2.1 The Wright-Fisher model 14

2.2.2 The Poisson random field 16

2.3 Time-Varying Feature Allocation Model 17

2.3.1 The WF-IBP 18

2.3.2 Prior parameters 20

2.3.3 Related models 21

2.4 Fixed-K approximation 22

2.4.1 Fixed-K MCMC inference 23

2.4.2 Approximation for large K . . . 25
2.5 Exact MCMC inference . . . 28
2.5.1 Gibbs and slice sampling . . . 29
2.5.2 Particle Gibbs and thinning . . . 30
2.6 Application: linear-Gaussian likelihood model . . . 33
2.6.1 Simulations and results . . . 35
2.6.2 Model misspecification . . . 37
2.7 Topic modeling application . . . 39
2.7.1 The WF-IBP topic model . . . 41
2.7.2 Posterior inference . . . 42
2.7.3 Topic model: simulation and results . . . 44
2.8 Discussion . . . 53
2.9 Appendix . . . 58
2.9.1 Appendix A. Proof of Theorem 1 . . . 58
2.9.2 Appendix B. Simulating from the model . . . 58
2.9.3 Appendix C. Derivation of full conditionals . . . 61

3 Scalable and Robust Bayesian Sampling 64
3.1 Introduction . . . 65
3.2 Relativistic Hamiltonian Dynamics . . . 66
3.2.1 Relativistic Hamiltonian Monte Carlo . . . 68
3.3 Relativistic Stochastic Gradient Markov Chain Monte Carlo . . . 69
3.3.1 Relativistic Stochastic Gradient Hamiltonian Monte Carlo . . . 70
3.3.2 Relativistic Stochastic Gradient Descent (with Momentum) . . . 71
3.3.3 A Stochastic Gradient Nosé-Hoover Thermostat for Relativistic Hamiltonian Monte Carlo . . . 73
3.4 Experiments . . . 74
3.4.1 Toy examples . . . 74
3.4.2 Neural Networks . . . 77
3.5 Discussion . . . 79

4 Multi-Task Adaptive Bayesian Linear Regression **81**

4.1 Introduction . . . 82

4.2 Background and related work . . . 83

4.2.1 Related work . . . 84

4.3 Multi-task Adaptive Bayesian Linear Regression . . . 86

4.3.1 Posterior Inference and Learning . . . 87

4.3.2 Relationship with GPs . . . 88

4.3.3 Computational Implications . . . 89

4.3.4 Transfer Learning Settings . . . 90

4.3.5 Online mode . . . 90

4.4 Experiments . . . 91

4.4.1 Experimental Set-up . . . 91

4.4.2 Transfer learning across parametrized quadratic functions . . 92

4.4.3 Transfer learning across OpenML data sets . . . 95

4.4.4 Tuning feedforward neural networks from heterogeneous signals 98

4.5 Discussion . . . 99

5 Bayesian Deep Learning for Population Genomic Data **101**

5.1 Introduction . . . 102

5.2 Related Work . . . 103

5.3 Methodology . . . 104

5.3.1 Feature Representation for Exchangeable Data . . . 105

5.3.2 Simulation-on-the-fly . . . 106

5.3.3 Likelihood-Free Inference Framework . . . 107

5.4 Population Genetics Application . . . 108

5.4.1 Recombination Hotspot Testing . . . 108

5.4.2 Recombination Hotspot Details . . . 109

5.5 Experiments . . . 111

5.5.1 Evaluation of Exchangeable Representation . . . 112

5.5.2 Evaluation of Simulation-on-the-fly . . . 113

5.5.3 Comparison to `LDhot` . . . 114

5.5.4 CEU data validation . . . 116

5.6 Discussion 117

6 Conclusions **120**

6.1 Future work 121

Acknowledgments

Firstly, I would like to express my sincere gratitude to Dario Spanò, Paul Jenkins and Yee Whye Teh. I have been extremely fortunate to have had them as my supervisors and will always be grateful for their advice, discussions and encouragement throughout the PhD. They have been dedicated to helping me grow and excel, and their guidance and support has been invaluable.

I was lucky to have some amazing collaborators and friends during my time at Oxford and Warwick. Thanks to the many people that helped through discussions during the writing of the papers composing this thesis, and especially to everyone in the Oxford statistical machine learning group for the collaborations and time together.

I would also like to thank Yun Song for making me feel part of such a great research group during my visit at UPenn. I am deeply grateful for his invaluable guidance and our enjoyable collaboration.

I am grateful to the research team at Amazon, Berlin, I had the fortune to work with during my internship. Special thanks to Rodolphe Jenatton, Cédric Archambeau, Matthias Seeger, Barbara Pogorzelska, and all the wonderful people I had to chance to meet during my time there.

I gratefully acknowledge the funding received towards my PhD from the EPSRC as part of the Oxford Warwick Statistics Programme.

Declarations

I hereby declare that, except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this or any other University. This dissertation is the result of my own work except where specifically indicated in the text.

- The work in Chapter 2 has been published in the *Journal of Machine Learning Research (JMLR)* [Perrone et al., 2017].
- The work in Chapter 3 has been published in the *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)* [Lu, Perrone et al., 2017] (joint first authors).
- The work in Chapter 4 has been accepted for publication by *Advances in Neural Information Processing Systems 31 (NIPS)*.
- The work in Chapter 5 has been accepted for publication by *Advances in Neural Information Processing Systems 31 (NIPS)*.

Valerio Perrone, October 2018

Abstract

In the last years we have witnessed an unprecedented increase in the volume and complexity of the available data. To reason about the world and address scientific questions, we need to build expressive and scalable models that can leverage this extraordinary amount of information. The Bayesian paradigm is an essential tool to formalize beliefs and quantify uncertainty, and is one of the cornerstones of machine learning. In this work, we develop new models for Bayesian machine learning that are able to tackle a number of challenges posed by large-scale data.

We first introduce a novel Bayesian nonparametric latent feature model for dependent data. Latent feature models aim to explain data in terms of a smaller set of hidden components which are responsible for the properties of the objects we observe. However, most latent feature models either fix the total number of features in advance or impose unrealistic exchangeability assumptions on the data. We overcome these challenges by constructing a novel time-evolving Bayesian nonparametric prior that learns the number of features from the data while modeling their time evolution explicitly.

In the second part of the thesis, we aim to combine the power of deep neural networks to learn useful representations with the ability of Bayesian learning to provide uncertainty estimates. To this end, we first design robust stochastic gradient algorithms for scalable inference in Bayesian neural network models. Then, we consider a simpler model consisting of a Bayesian linear regression layer built on top of a deep neural network. We demonstrate the benefits of this approach in a Bayesian optimization setting, scaling to millions of datapoints and transferring knowledge across tasks. Finally, we establish a deep learning framework for large-scale Bayesian inference and apply it to fully harness raw population genomic data.

1

Introduction

Bayesian models are a fundamental tool to deal with uncertainty in a stochastic world. Bayesian statistics makes it possible to formalize beliefs and update them as data is observed. Initial beliefs are encoded in a prior, which consists of a probability distribution on the parameter of interest, while Bayes rule provides a way to combine these beliefs with the likelihood of the observed data to obtain a posterior distribution. This is a natural paradigm that has seen widespread success in a broad range of domains.

The volume of data we have access to has been growing exponentially in recent years, calling for models and inference techniques that can rise to the challenge. On one hand, more flexible models are needed to adapt automatically to the complexity of the observed data. The dimensionality of the parameter space should not be fixed a priori but learned from the data, allowing for open-ended degrees of freedom. Bayesian nonparametric statistics is an elegant approach to achieve this. On the other hand, while rich and flexible models are desirable, their practical use is hindered if their computational tractability is limited. Current datasets call for models for which inference is easy to perform at large scale.

Deep learning has emerged as an extremely powerful tool to capture complex functional relationships as well as to scale to massive datasets, leading to countless successes and era-defining technologies. The black-box nature of many deep learning methods, however, is at odds with the fully-principled Bayesian approach. Model interpretability is often sacrificed for the sake of computational speed, and point

estimates of model parameters are typically provided instead of posterior probability distributions. This work aims to contribute to the field of machine learning with a set of new Bayesian models that can scale up to large datasets while offering a principled way to represent uncertainties.

Throughout this thesis, a special attention will be devoted to the trade-off between model richness and computational tractability. A large amount of work in the Bayesian and machine learning communities has been dedicated to either one of these goals. This work strives to pursue both directions by introducing flexible Bayesian models for machine learning that are amenable to tractable and scalable inference. In a broader perspective, this thesis seeks to make a step towards a wider use of Bayesian methodology for applications in large-scale machine learning.

1.1 Bayesian nonparametrics

A probabilistic representation of uncertainty is crucial to several tasks, from scientific modelling to decision making. Ideally, we would like our models to output a reliable measure of confidence together with their predictions, in such a way that the larger the uncertainty over the results, the lower the output confidence. There exist multiple sources of uncertainty, which can be grouped into two main categories. The first one is epistemic uncertainty, namely uncertainty over the model parameters that best explain the observed data. This can usually be reduced by gathering more data. The second one is aleatoric uncertainty, which is instead due to the noise inherent in the data and does not depend on the chosen model. For instance, noise can be associated to the process in which the data was acquired, e.g., through imprecise measurements.

The Bayesian framework offers a principled solution to quantify both sources of uncertainty. In this setting, parameters are assumed to be random variables following a prior probability distribution that reflects our initial beliefs. At the heart of Bayesian inference lies the update of such prior into a posterior distribution that accounts for the likelihood of the observed data. This offers a natural way to reduce epistemic uncertainty as more data is collected and the information about the prior is updated. Aleatoric uncertainty can also be modelled in the Bayesian

setting. To this end, the likelihood can be enriched with a (typically Gaussian) noise term, which is inferred to estimate the amount of noise inherent in the data. The overall predictive uncertainty yielded by the Bayesian model can be then effectively represent a combination of both epistemic and aleatoric uncertainty. These notions of uncertainty underlie the models and results presented in this thesis.

One limit of standard Bayesian approaches is that the dimensionality of the parameter space is fixed and there is a finite number of parameters. As phenomena become increasingly complex, however, more sophisticated models are required to describe them. This calls for a shift from a parametric to a nonparametric approach, wherein the size of the parameter space is not limited a priori but adapts to the complexity of the data automatically. A common assumption of most Bayesian nonparametric models is exchangeability, which we briefly review in the next section before introducing the concept of completely random measures.

1.1.1 Exchangeability

A finite sequence of random variables is finitely exchangeable if any permutation of the random variables has the same joint distribution. Intuitively, this implies that the order in which data arrives is irrelevant. Formally, a sequence of random variables $X_1, \ldots, X_n$ is exchangeable if

$$\mathbb{P}\Big(X_1, \ldots, X_n\Big) = \mathbb{P}\Big(X_{\sigma(1)}, \ldots, X_{\sigma(n)}\Big),$$

for all permutations σ of the indices $\{1, \ldots, n\}$. An infinite sequence is infinitely exchangeable if exchangeability holds for any finite subset. For any infinitely exchangeable sequence of random variables $X_1, \ldots, X_n$, de Finetti's representation theorem [de Finetti, 1937] reveals the existence of an underlying mixing measure B such that

$$P(X_1, \ldots, X_N) = \int \prod_{i=1}^{N} P(X_i|\theta) B(d\theta),$$

where θ is a parameter conditionally on which $X_1, \ldots, X_n$ are independent.

Despite being very common and simplifying inference, the exchangeability assumption is not always realistic. In many real-world problems, observations come

in distinct groups so that dependencies are stronger within the same group. This is the key idea behind the model we introduce in Chapter 2, where we develop a novel Bayesian nonparametric prior that relaxes the undesirable exchangeability assumption to model time-evolving data. On the other hand, data can also exhibit partial forms of exchangeability. For instance, consider the case of genome sequences sampled from individuals of an unstructured population. In this setting, while permutations of individual DNA blocks would completely alter the profile of the sequence, the entire DNA sequences from different individuals are typically exchangeable for the purposes of population-level inference. The idea of tailoring existing models to harness this form of exchangeability is explored in Chapter 5, with a direct application to population genetics.

1.1.2 Completely random measures

Completely random measures make it possible to define models with open-ended degrees of freedom. A random measure is defined as a random variable whose values are measures on a measurable space (Ω, Σ). A completely random measure B over a measurable space (Ω, Σ) is a random measure that assigns independent masses to disjoint subsets of Ω. Any positive completely random measure is uniquely characterized by a certain Levy measure on $\Omega \times \mathbb{R}^+$ [see Kingman, 1967]. Denote by c a positive function over Ω (concentration function) and by B_0 a fixed measure on Ω (base measure), and assume the base measure B_0 is continuous.

The beta process is an example of completely random measure, as well as one of the pillars of Bayesian nonparametrics. Formally, a beta process $\mathrm{B} \sim BP(c, B_0)$ on Ω is a completely random measure uniquely characterized by the Levy measure

$$\nu(d\omega, dx) = B_0(d\omega)c(\omega)x^{-1}(1-x)^{c(\omega)-1}dx,$$

where $x \in [0, 1]$ and $\omega \in \Omega$. This completely random measure can be represented via a Poisson process. In order to draw $B \sim BP(c, B_0)$, draw a set of points $(\omega_i, x_i) \in \Omega \times [0, 1]$ from a Poisson point process with base measure $\nu(d\omega, dx)$ and

let

$$B = \sum_{k=1}^{\infty} x_k \delta_{\omega_k},$$

where $\{\omega_i\}$ are the atoms of the measure B and $\{x_i\}$ their respective weights. Chapter 2 demonstrates how these mathematical objects can be leveraged to infer the dimensionality of the parameter space from the data, without having to bound it in advance.

1.2 Deep learning

A drawback of Bayesian nonparametric models is that inference can be challenging. Sophisticated samplers or finite approximations schemes typically need to be adopted, limiting the widespread use of Bayesian nonparametric approaches in practice. In this section we review deep learning, an alternative class of methods that can flexibly adapt to complex data.

Deep learning is one of the most successful machine learning paradigms in the last years, having broken records in a broad range of applications, such as computer vision and speech recognition. One of the core reasons for its success lies in its scalable and out-of-the-box nature, which makes it particularly suitable for data-intensive environments.

The building block of deep learning methods are neural networks. Neural networks can be seen as models defining a mapping, which is typically highly non-linear, from an input to an output. In particular, feed-forward neural networks define a nonlinear map $f_z(x)$ as follows. For the input vector x, consider the L-layer feed-forward transformation parametrized by the weight matrices $\{Z_l\}_{l=1}^{L}$:

$$f_z(x) = a_L \left(Z_L a_{L-1} \left(\dots Z_2 a_1 \left(Z_1 x \right) \dots \right) \right).$$

Common choices for the non-linear activation functions $a_1, \dots, a_L$ are `tanh` functions or `ReLU` (rectified linear units, i.e. $a(x) = max(0, x)$). In words, feed-forward neural networks generally consists of a sequence of inner products and non-linear

transformations.

Several architectures have been designed to exploit specific properties of input data. To this end, convolutional neural networks (CNNS) have been developed [LeCun et al., 1989] as popular deep learning tools for images. This class of models applies convolutional and pooling layers sequentially, followed by final inner product layers as in feed-forward neural networks. A convolutional layer is a linear transformation preserving local translation invariance in the input, while pooling layers reduce the dimension of the output of convolutions.

Neural networks can also be thought of as powerful functional approximators that are able to represent functions of increasing complexity. As problems become more complex, deeper neural networks with more hidden layers can be used. In a sense, deep learning shares with Bayesian nonparametrics the goal of learning from data exhibiting a high degree of complexity. Compared to complex Bayesian inference, training deep learning models is often simple and scales to a large number of datapoints, but point estimates of the parameters of interest are usually provided instead of posterior distributions. Bayesian neural networks aim at combining the best of both worlds, placing a prior distribution on the neural network parameters and learning full posterior distributions. This and some alternative classes of Bayesian deep learning models will be explored in Chapter 4 and 5.

1.3 Inference

One of the most popular techniques for Bayesian inference is Markov Chain Monte Carlo (MCMC) [Andrieu et al., 2003], allowing for approximate inference with unnormalized and high-dimensional distributions. The method relies on Markov chain theory to obtain samples from a target distribution $g(x)$, and is also applicable when $g(x)$ is known only up to a normalization constant.

The core idea is to define a transition distribution $p(x_{t+1}|x_t)$ such that the target $g(x)$ is stationary for the resulting Markov chain. If samples are sequentially drawn from $p(x_{t+1}|x_t)$, under a set of assumptions the distribution $p(x_t)$ converges to the target $g(x)$ for $t \to \infty$.

A few drawbacks of MCMC include that it generates correlated samples and

requires a burn-in phase to make $p(x_t)$ converge to $g(x)$. The first shortcoming can be mitigated by thinning, namely using only one sample every k for the Monte-Carlo estimation. The second issue is handled ignoring the first n samples, provided that n is sufficiently large. The most popular MCMC methods are Metropolis-Hastings (MH) [Hastings, 1970] and Gibbs sampling [Geman and Geman, 1984], which we review, before describing Particle Filters.

Metropolis-Hastings In Metropolis-Hastings (MH), the $t+1$-th sample is drawn from a proposal distribution $q(x_{t+1}|x_t)$, after which it is accepted with probability $min\left(\frac{g(x_{t+1})q(x_t|x_{t+1})}{g(x_t)q(x_{t+1})|x_t)}, 1\right)$. If the proposal distributions are not properly designed, vanilla MH may exhibit a low acceptance rate and remain stuck in high-probability isolated regions without traversing the remaining space. This is a consequence of the proposal distribution making local moves. Several alternatives samplers have been proposed to address these issues, with one of the most popular algorithms being Hamiltonian Monte Carlo (HMC), which we review in Chapter 3.

Gibbs sampling In Gibbs sampling, the idea is to generate each n-dimensional sample $x = \{x(1), x(2), \ldots, x(n)\}$ by sampling, in turn, every component $x(i)$ conditioning on the latest values of the others $x \setminus \{x(i)\}$ according to the target distribution $g(x)$. This requires the full conditional distributions to be available. For instance, for a two-dimensional state $x = \{a, b\}$ and a target distribution $g(a, b)$, a_t is sampled from $g(a|b_{t-1})$ and b_t is sampled from $g(b|a_t)$. Several variations of Gibbs sampling have been developed. For example, collapsed Gibbs Sampling performs Gibbs sampling on a subset of components, marginalizing out the remaining ones. Another example is blocked Gibbs Sampling that samples group of variables together. This approach is at the base of the sampler we design in Chapter 2.

Particle Filters Consider a Hidden Markov Model setting, where u_t is an input, or action, that influences the next hidden states x_{t+1} and observations z_{t+1}. The goal in temporal models is to estimate the probability density function $b(x_t) = p(x_t|z_{1:t}, u_{1:t})$. The Bayes filter recursively computes the belief at time $t+1$, starting from the belief at time t, the last observation z_{t+1}, and the last action u_{t+1} via

$b(x_{t+1}) = \nu p(z_{t+1}|x_{t+1}) \int_{x_t} p(x_{t+1}|x_t, u_{t+1}) b(x_t) dx_t$, being ν a normalization constant.

The idea is to represent the belief by a set of weighted samples, which are referred to as particles. The resulting algorithm is called Particle Filter, or sequential Monte-Carlo [Andrieu et al., 2010]. Given M weighted samples $\{(x_t^{(i)}, w_t^{(i)})\}$ distributed as $b(x_t)$, a new observation z_{t+1} and a new action u_{t+1}, the particle filter generates a new weighted sample set that approximates $b(x_{t+1})$. The algorithm consists of three steps:

- Prediction: sample a fresh set of samples $x_{t+1}^{(i)}, i = 1, \ldots, M$, from a proposal distribution $g(x_{t+1}|x_t^{(i)}, z_{t+1}, u_{t+1})$.
- Weighting: give each sample $x_{t+1}^{(i)}$ the following weight:

$$w_{t+1}^{(i)} = w_t^{(i)} \frac{p(z_{t+1}|x_{t+1}^{(i)}) p(x_{t+1}^{(i)}|x_t^{(i)}, u_{t+1})}{g(x_{t+1}^{(i)}|x_t^{(i)}, z_{t+1}, u_{t+1})}.$$

- Resampling: resample the particles with replacement with probability proportional to $w_{t+1}^{(i)}$, and set the weights to 1.

A common simplification, which we will also apply in Chapter 2, is the bootstrap filter, whereby $g(x_{t+1}|x_t^{(i)}, z_{t+1}, u_{t+1}) = p(x_{t+1}|x_t^{(i)}, u_{t+1})$ and the weight simplifies $w_{t+1}^{(i)} = w_t^{(i)} p(z_{t+1}|x_{t+1}^{(i)})$.

ABC In several domains, such as the complex models from population genetics we consider in Chapter 5, the likelihood term is either analytically or computationally intractable. A range of likelihood-free inference methods have been developed to address this, with the most popular one being Approximate Bayesian Computation (ABC) [Beaumont et al., 2002].

Suppose x_{obs} is the observed dataset, S a summary statistic, d a distance metric, and θ the parameter of interest. Assume it is possible to simulate from the likelihood model, namely to generate datapoints from the likelihood $P(x \mid \theta)$, conditioning on θ being drawn from the prior $\pi(\theta)$. The algorithm for vanilla rejection ABC is as follows. Denoting by i each simulated dataset, for $i = 1 \ldots N$,

1. Simulate $\theta^{(i)} \sim \pi(\theta)$ and $x^{(i)} \sim P(x \mid \theta^{(i)})$

2. Keep $\theta^{(i)}$ if $d(S(x^{(i)}), S(x_{obs})) \leq \epsilon$,

where $\epsilon > 0$. The output provides an empirical estimate of the posterior. Two key results regarding ABC make it an attractive method for Bayesian inference. First, it has an asymptotic guarantee in that, as $\epsilon \to 0$, $N \to \infty$, and if S is sufficient, the estimated posterior converges to the true posterior. Second, a variant of ABC which injects noise into the summary statistic function is calibrated. Note that ABC is notoriously difficult to perform diagnostics on without the ground truth posterior as many factors could contribute to a poor posterior approximation: poor choice of summary statistics, incorrect distance metric, insufficient number of samples, or large ϵ. Chapter 5 introduces an approach based on deep learning to overcome these drawbacks.

1.4 Outline and contributions of thesis

This thesis consists of four main parts. In Chapter 2, we introduce a novel Bayesian nonparametric model for time-dependent data. In a feature allocation model, each data point depends on a collection of unobserved latent features. For instance, we might classify a corpus of texts by describing each document via a set of topics; the topics then determine a distribution over words for that document. In a Bayesian nonparametric setting, the Indian Buffet Process (IBP) is a popular prior model in which the number of topics is unknown a priori. However, the IBP is static in that it does not account for the change in popularity of topics over time. We present the *Wright-Fisher Indian Buffet Process* (WF-IBP), a probabilistic model for collections of time-stamped documents. By adapting the Poisson random field model from population genetics, we derive a stochastic process with appealing properties including that each feature popularity evolves independently as a diffusion and marginal observations at a fixed timepoint are given by the IBP.

In Chapters 3, 4 and 5, we leverage deep neural networks to learn flexible data representations within a Bayesian framework. In particular, Chapter 3 focuses on the problem of scaling up stochastic gradient MCMC, a family of techniques to

sample from complex models such as Bayesian neural networks while scaling to large datasets. Specifically, we introduce *Relativistic Monte Carlo*, a robust alternative to Hamiltonian Monte Carlo that limits the speed of the parameter changes to better control the stability of the sampler. In Chapter 4, we design a model consisting of a deep neural network whose last layer is Bayesian and refer to it as *Adaptive Bayesian Linear Regression* (ABLR). This model can also be thought of as a simple Bayesian linear regression applied to a feature transformation of the input data, which is learned via a shared neural network. We explore the appeal of the model in a Bayesian optimization setting, showing that the Bayesian layer allows for full uncertainty estimates while the shared neural network is able to transfer knowledge across related tasks. In Chapter 5, we apply Bayesian and deep learning ideas to inference with population genetic data. While high throughput DNA sequencing is becoming the norm, extracting useful features from genomic data is challenging. We devise a novel framework to improve on existing likelihood-free techniques and work with raw genomic data directly. Finally, in Chapter 6 we outline our conclusions as well as some directions for future work.

2

Bayesian Nonparametric Dynamic Feature Models

In this chapter we present the *Wright-Fisher Indian buffet process* (WF-IBP), a probabilistic model for time-dependent data assumed to have been generated by an unknown number of latent features. This model is suitable as a prior in Bayesian nonparametric feature allocation models in which the features underlying the observed data exhibit a dependency structure over time. More specifically, we establish a new framework for generating dependent Indian buffet processes, where the Poisson random field model from population genetics is used as a way of constructing dependent beta processes. Inference in the model is complex, and we describe a sophisticated Markov Chain Monte Carlo algorithm for exact posterior simulation. We apply our construction to develop a nonparametric focused topic model for collections of time-stamped text documents and test it on the full corpus of NIPS papers published from 1987 to 2015.

2.1 Introduction

The Indian buffet process (IBP) [Griffiths and Ghahramani, 2011] is a distribution for sampling binary matrices with any finite number of rows and an unbounded number of columns, such that rows are exchangeable while columns are independent. It is used as a prior in Bayesian nonparametric models where rows represent objects and columns represent an unbounded array of features. In many settings the prevalence of features exhibits some sort of dependency structure over time and modeling data via a set of independent IBPs may not be appropriate. There has been previous work dedicated to extending the IBP to dependent settings [e.g., Williamson et al., 2010a; Zhou et al., 2011; Miller et al., 2012; Gershman et al., 2015]. In this chapter, we present a novel approach that achieves this by means of a particular time-evolving beta process, which has a number of desirable properties and is better-suited for a different range of applications.

For each discrete time $t_0, \ldots, t_T$ at which the data is observed, denote by Z_t the feature allocation matrix whose entries are binary random variables such that $Z_{ikt} = 1$ if object i possesses feature k at time t and 0 otherwise. Denote by $X_k(t)$ the probability that $Z_{ikt} = 1$, namely the probability that feature k is active at time t, and by $X(t)$ the collection of these probabilities at time t. The idea is to define a prior over the stochastic process $\{X(t)\}_{t\geq 0}$ which governs its evolution in continuous time. In particular, for each feature k, $X_k(t)$ evolves independently, while features are born and die over time. This is a desirable property in several applications such as in topic modeling, where at some point in time a new topic may be discovered (birth) or forsaken (death). Our model benefits from these properties while retaining a very simple prior where sample paths are continuous and Markovian. Finally, we show that our construction defines a time-dependent beta process from which the two-parameter generalization of the IBP is marginally recovered for every fixed time t [Ghahramani et al., 2007].

The stochastic process we use is a modification of the so-called Poisson Random Field (PRF), a model widely used in population genetics [e.g., Sawyer and Hartl, 1992; Hartl et al., 1994; Bustamante et al., 2001, 2003; Williamson et al., 2005; Boyko et al., 2008; Gutenkunst et al., 2009; Amei and Sawyer, 2010, 2012]. In this

setting, new features can arise over time and each of them evolves via an independent Wright-Fisher (W-F) diffusion. The PRF model describes the evolution of feature probabilities within the interval $[0, 1]$, allows for flexible boundary behaviours and gives access to several off-the-shelf results from population genetics about quantities of interest, such as the expected lifetime of features or the expected time feature probabilities spend in a given subset of $[0, 1]$ [see Ewens, 2004].

We apply the WF-IBP to a topic modeling setting, where a set of time-stamped documents is described using a collection of latent topics whose probabilities evolve over time. The WF-IBP prior allows us to incorporate time dependency into the focused topic model construction described in Williamson et al. [2010b], where the IBP is used as a prior on the topic allocation matrix determining which topics underlie each observed document. As opposed to several existing approaches to topic modeling, which require specifying the total number of topics in the corpus in advance [see for instance Blei et al., 2003], adopting a nonparametric approach saves expensive model selection procedures [such as the one described in Griffiths and Steyvers, 2004]. This is also reasonable in view of the fact that the total number of topics in a corpus is expected to grow as new documents accrue. Most existing nonparametric approaches to topic modeling are not designed to capture the evolution of the popularity of topics over time and may thus not be suitable for corpora that span large time periods. On the other hand, existing nonparametric and time-dependent topic models are mostly based on the Hierarchical Dirichlet Process (HDP) [Teh et al., 2006], which implicitly assumes a coupling between the probability of topics and the proportion of words that topics explain within each document. This assumption is undesirable since rare topics may account for a large proportion of words in the few documents in which they appear. Our construction inherits from the static model presented in Williamson et al. [2010b] the advantage of eliminating this coupling. Moreover, it keeps inference straightforward while using an unbounded number of topics and flexibly capturing the evolution of their popularity continuously over time.

Section 2.2 introduces the beta process construction of the IBP and the PRF, providing the background for Section 2.3 where we modify the PRF to construct

the WF-IBP, our novel time-varying feature allocation model. Section 2.4 describes a fixed-K approximation and an intuitive inference scheme that is built upon in Section 2.5 to develop an exact MCMC algorithm for posterior inference with the WF-IBP. Section 2.6 combines the model with a linear-Gaussian likelihood and evaluates it on a range of synthetic data sets. Finally, Section 2.7 illustrates the application of the WF-IBP to topic modeling and presents results obtained on both synthetic data and on the real-world data set consisting of the full text of papers from the NIPS conferences between the years 1987 and 2015.

2.2 Background

Before introducing our time-varying feature allocation model, we first review its building block, namely the PRF. We show how the beta process connects this model with the IBP, and in the next section we adjust the PRF to develop a time-dependent extension of the two-parameter IBP.

2.2.1 The Wright-Fisher model

The starting point of the PRF is the Wright-Fisher (W-F) model from population genetics [Ewens, 2004], which we briefly summarize here. Consider a finite population of organisms of size G such that $i \in \{0, 1, \ldots, G\}$ individuals have the mutant version of a gene at generation k, while the rest has the non-mutant variant. Assume that each individual produces an infinite number of gametes such that the gametes yielded by a non-mutant become mutant with probability μ_G and, conversely, those yielded by a mutant become non-mutant with probability β_G. Finally, assume that the next generation of G individuals is formed by simple random sampling from this infinite pool of gametes. The evolution of the number $Y^G(k)$ of mutant genes at time k is described by a Markov chain on the discrete space $\{0, \ldots, G\}$. The transition probability p_{ij} of switching from i mutants (at time k) to j mutants (at

time $k+1$) is given by the following binomial sampling formula:

$$p_{ij} = \binom{G}{j}(\Psi_i)^j(1-\Psi_i)^{G-j},$$
$$\Psi_i = \frac{i(1-\beta_G)+(G-i)\mu_G}{G}.$$

Assume the initial state is $Y^G(0) = y_0$ and denote the resulting Markov chain by $Y^G = (Y^G(k))_{k=1,2,\ldots} \sim \text{W-F}^G(\mu_G, \beta_G)$. Notice that, if $\mu_G = 0$ and/or $\beta_G = 0$, the states 0 and/or G are absorbing states that respectively correspond to the extinction and fixation of the mutation.

A continuous-time diffusion limit of the W-F model can be obtained, by rescaling time as $t = k/G$, and taking $G \to \infty$. The Markov chain $Y^G(\lfloor Gt \rfloor)/G$ converges to a diffusion process on $[0,1]$ [see Ethier and Kurtz, 1986; Sawyer and Hartl, 1992] which obeys the one-dimensional stochastic differential equation

$$dX(t) = \gamma(X(t))dt + \sigma(X(t))dB(t),$$

where

$$\gamma(x) = \frac{1}{2}[\mu(1-x) - \beta x], \tag{2.1}$$
$$\sigma(x) = \sqrt{x(1-x)}, \tag{2.2}$$

with some initial state $X(0) = x_0$, over the time interval $t \in [0,T]$, with rescaled parameters $\mu = \lim_{G\to\infty} 2G\mu_G$, $\beta = \lim_{G\to\infty} 2G\beta_G$, and with $B(t)$ denoting a standard Brownian motion. The terms $\gamma(x)$ and $\sigma(x)$ are respectively referred to as the drift term and the diffusion term. Denote the diffusion process as $X \sim \text{W-F}(\mu, \beta)$.

When $x(t) \to 0$ (respectively $x(t) \to 1$), then the diffusion term tends to 0 while the drift term tends to $\frac{\mu}{2}$ (respectively $-\frac{\beta}{2}$), preventing absorption at 0 or 1 provided that $\mu > 0$ (respectively $\beta > 0$). Otherwise, 0 is an absorbing extinction state (respectively, 1 is an absorbing fixation state). Moreover, if both $\mu, \beta > 0$ then the diffusion is ergodic and has a stationary distribution that is a Beta(μ, β).

As there exists no closed form expression for its transition function, simu-

lating from the W-F diffusion requires non-trivial computational techniques. The method we used is outlined in Dangerfield et al. [2012], a stochastic Taylor scheme tailored to the W-F diffusion. A novel alternative approach that allows for exact simulation has very recently been proposed by Jenkins and Spanò [2017]. Finally, note that simulating from the W-F diffusion implies doing so for a given diffusion time unit Δt. While in population genetics diffusion time is related to the population size, in different applications it can be used to regulate the degree of sample path variance, which is linear in time and equal to $\mathrm{Var}(x(1-x)\Delta t)$.

2.2.2 The Poisson random field

The W-F model describes the evolution of a gene at one particular site. The PRF generalizes it to modeling an infinite collection of sites, each of which evolves independently according to the W-F model. As before, we start with a model with a population of finite size G, before taking the diffusion limit as $G \to \infty$. For a site i at which some individuals carry the mutant gene, denote by $X_i(k)$ the fraction of mutants in generation k. Each site evolves independently according to the W-F$^G(0,0)$ model. Further suppose that at each generation k a number of mutations $M \sim \mathrm{Poisson}(\nu_G)$ arise in new sites with indices $j_1, j_2, \ldots j_M$. ν_G is referred to as the immigration parameter of the PRF. Assume that each of the new mutations occurs at a new site in a single individual, with initial frequency $X_{j_m}(k) = 1/G$. Subsequently, each new process $X_{j_m}(k+1), X_{j_m}(k+2), \ldots$ evolves independently according to the W-F$^G(0,0)$ model as well (Figure 2.1). As with pre-existing mutant sites, each process eventually hits one of the boundaries $\{0,1\}$ and stays there (we say that the mutation is extinct/has been fixed).

Consider the limit $G \to \infty$, so that after the same rescaling $t = k/G$ of time as in Section 2.2.1 each site evolves as an independent W-F diffusion $X_i \sim$ W-F$(0,0)$. We also assume that $\nu_G \to \alpha$ as $G \to \infty$. This means that in the diffusion time scale the immigration rate is $G\nu_G \to \infty$, which suggests that the number of sites with mutant genes should explode. However, the initial frequency of each diffusion is $1/G \to 0$ as $G \to \infty$, and moreover 0 is an absorbing state. It can be shown [Sawyer and Hartl, 1992; Amei and Sawyer, 2010] that only $O(G^{-1})$

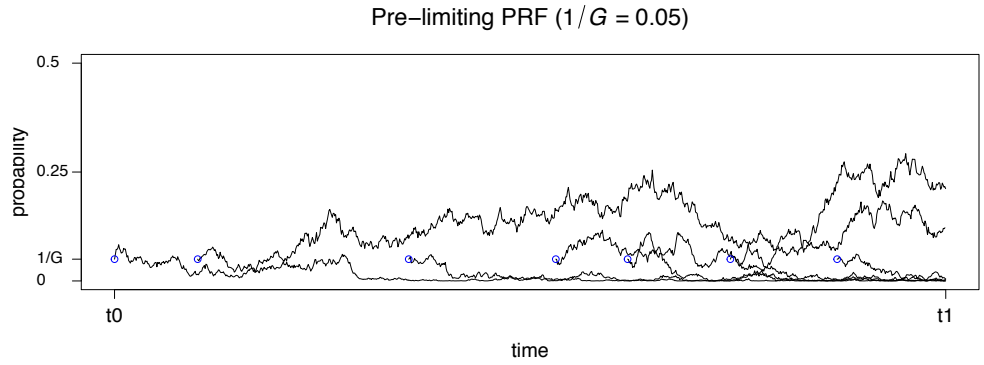


Figure 2.1: Evolution of mutant sites over time in the pre-limiting PRF model. The blue circles indicate mutations arising at a new site.

of the newborn processes are not almost immediately absorbed. Therefore, there is a balance between the infinite number of newborn mutations and the infinite number of them going extinct in the first few generations, in such a way that the net immigration rate is $O(G\nu_G \times G^{-1}) = O(\alpha)$, and hence the limiting stationary measure is nontrivial. Provided that we remove from the model all sites whose frequency hits either the boundary 1 or 0, Sawyer and Hartl [1992] prove that the limiting distribution of the fractions of mutants in the interval $[0, 1]$ is a Poisson random field with mean density

$$\alpha x^{-1} dx. \tag{2.3}$$

Interestingly, the rate measure of the PRF coincides with the distribution of weights in the one-parameter beta process. This means that at equilibrium the number of sites whose frequencies $X_i(t)$ are in any given interval $(a, b]$ is Poisson distributed with rate $\alpha \int_a^b x^{-1} dx$, and these are independent for nonoverlapping intervals. Integrating (2.3) over $[0, 1]$ shows that the number of mutations in the population that has not been fixed or gone extinct is infinite. However, most mutations are present in a very small proportion of the population.

2.3 Time-Varying Feature Allocation Model

The derivation of the PRF in the previous section shows that, as long as sites reaching frequency 1 or 0 are removed from the model, the equilibrium distribution of the PRF is related to the one-parameter beta process. In this section we generalize

the PRF so that it is better adapted to applications in feature allocation modeling. Specifically, we identify mutant sites with features, and identify the proportion of the population having the mutant gene with the probability of the feature occurring in a data observation. The PRF can be then used in a time-varying feature allocation model whereby features arise at some unknown time point, change their probability smoothly according to a W-F diffusion process and eventually die when their probability reaches zero.

2.3.1 The WF-IBP

Recall from the previous section that mutant sites whose frequency hits 1 are removed from the PRF model. This means that features with high probability of occurrence can be removed from the model instantaneously, which does not make modeling sense. Instead, one expects a feature probability to change smoothly and to be removed from the model only once its probability of occurrence is *small*. A simple solution to this conundrum is to prevent 1 from being an absorbing state by using instead a W-F$(0, \beta)$ diffusion with $\beta > 0$. This is a departure from Sawyer and Hartl [1992], due to the differing modeling requirements of genetics versus feature allocation modeling. At the same time, both models let features disappear once their probability gets to 0, which is suitable from a feature allocation perspective and, as we now see, allows for a nontrival equilibrium mean density.

We shall denote the modified stochastic process as PRF(α, β). The following theorem derives the equilibrium mean density of PRF(α, β), with proof given in Appendix A:

Proposition 1 *The equilibrium mean density of the* PRF(α, β) *is*

$$l(x) = \alpha x^{-1}(1-x)^{\beta-1}dx. \tag{2.4}$$

In other words, the mean density of the PRF(α, β) is the Lévy measure of the two-parameter beta process, with the immigration rate α identified with the mass parameter, and β identified with the concentration parameter of the beta process. When $\beta = 1$ the one-parameter beta process is recovered. We assume that the initial distribution of PRF(α, β) is its equilibrium distribution, that is, a Poisson

random field with mean density (2.4), so that the marginal distribution of the PRF at any point in time is the same.

We will now make the connection more precise by specifying how a PRF can be used in a time-varying feature allocation model. Denote by $X_k(t)$ the probability of feature k being active at time t and define our PRF as the stochastic process $X := \{X_k(t)\}$. Assume that at a finite number of time points $t = t_0, \ldots, t_T$ there are N_t objects whose observable properties depend on a potentially infinite number of latent features. Let D_{it} be the observation associated with object $i = 1, \ldots, N_t$ at time $t = t_0, \ldots, t_T$. Consider a set of random feature allocation matrices Z_t such that entry Z_{ikt} is equal to 1 if object i at time t possesses feature k, and 0 otherwise. Let $Z := \{Z_{ikt}\}$. Finally, let ρ_k be some latent parameters of feature k and $\rho = \{\rho_k\}$ be the set of all feature parameters. Our complete WF-IBP model is given as follows.

$$\begin{aligned} X &\sim \text{PRF}(\alpha, \beta), \\ Z_{ikt} \mid X &\overset{ind}{\sim} \text{Bernoulli}(X_k(t)), \\ \rho_k &\overset{iid}{\sim} H, \\ D_{it} \mid \rho, Z_{it} &\overset{ind}{\sim} F(\{\rho_k : Z_{ikt} = 1\}), \end{aligned} \tag{2.5}$$

where $i = 1, \ldots, N_t$, $t = t_0, \ldots, t_T$ and $k = 1, 2, \ldots, H$ is the prior distribution for feature parameters, and where $F(\rho)$ is the observation model for an object with a set of features with parameters ρ.

Since the feature probabilities X have marginal density (2.4), at each time t the feature allocation matrix Z_t has marginal distribution given by the two-parameter Indian buffet process [Thibaux and Jordan, 2007]. Further, since X varies over time, the complete model is a time-varying Indian buffet process feature allocation model. The corresponding de Finetti measure would then be a time-varying beta process. More precisely, this is the measure-valued stochastic process $G = \{G(t)\}$ where

$$G(t) = \sum_{k=1}^{\infty} X_k(t) \delta_{\rho_k},$$

which has marginal distribution given by a beta process with parameters α, β and base distribution H. We denote the distribution of G as $\text{WFBP}(\alpha, \beta, H)$. We can also express the feature allocations using random measures as well. In particular, let

$$B_{it} = \sum_{k=1}^{\infty} Z_{ikt}\delta_{\rho_k}$$

be a Bernoulli process $\text{BeP}(G(t))$ with mean measure given by the beta process $G(t)$ at time t. An equivalent way to express our model (2.5) using the introduced random measures is then

$$\begin{aligned} G &\sim \text{WFBP}(\alpha, \beta, H), \\ B_{it} \mid G &\sim \text{BeP}(G(t)), \\ D_{it} \mid B_{it} &\sim F(B_{it}), \end{aligned}$$

where WFBP denotes our time-varying beta process, and we have used $F(B)$ to denote the same observation model as before, but with B being a random measure with an atom for each feature, and whose location is the corresponding feature parameter. In the following, we use the notation introduced in (2.5) instead of in terms of beta and Bernoulli processes for simplicity. A detailed description of how to simulate from the model is given in Appendix B, while in the next section with discuss the role of the model hyperparameters more closely.

2.3.2 Prior parameters

Extending the IBP, the model we propose naturally includes its hyper-parameters. As in the two-parameter IBP, α and β separately control the distribution of the number of features per object and the total number of features. In particular, the latter parameter can be thought of as controlling the feature "repulsiveness", being inversely proportional to the degree of feature sharing across objects. As the role of such prior parameters has been extensively studied for the IBP, in our experiments we will focus on the ability of the PRF-IBP to model time dependencies. To this end, it is important to note that the discrete time points $t_0, \ldots, t_T$ at which the observations are given influence the number of time units for which the W-F diffu-

sions should be simulated. This introduces an extra hyper-parameter that regulates the strength of the time-dependency or accounts for gaps of varying sizes between successive observations. In population genetics, the diffusion time is related to the population size. In general, this parameter regulates the degree of sample path variance, which is linear in time and equal to $\mathrm{Var}(x(1-x)\Delta t)$. In applications such as topic modeling, batches of documents are typically observed at a range of frequencies (daily in the case of newspapers, yearly in the case of conference papers and so on), so that the time hyperparameter can be set to account for the different degrees of dependence. For instance, the time parameter can be chosen for the dataset at hand according to the expected life time of features [see Chapter 15 of Karlin and Taylor, 1981].

2.3.3 Related models

The WF-IBP fits a line of research that aims at introducing dependency structures into the IBP. A number of these extensions are designed to drop the exchangeability assumption from the IBP by coupling the rows and columns of the feature allocation matrix, and are thus orthogonal to our work [see Zhou et al., 2011; Miller et al., 2012; Gershman et al., 2015]. What we are rather interested in achieving is partial exchangeabilty, whereby objects can be permuted independently at each time point without changing the probability of the process. This is necessary for time-dependent topic models where each time has a different set of documents and there is no correspondence between documents at different times.

One example that is more closely related to our model is the dependent Indian buffet process (dIBP) [Williamson et al., 2010a], which can also be applied to the partially exchangeable case. The dIBP uses a hierarchical Gaussian process to introduce couplings between features and items in such a way that, for an appropriate choice of the kernel, items can be permuted independently at each time. Although both the dIBP and the WF-IBP try to achieve the same goal, their methodologies are substantially different: while the former introduces dependencies at the feature-matrix level, the latter does so at the beta-process level. The construction of a dependent beta process represents a significantly different

research direction, and is indeed anticipated as future work in Williamson et al. [2010a]. First, an important consequence is that the WF-IBP prior describes the evolution of feature probabilities explicitly, whereas the dIBP fixes them and effectively describes a time-evolving Bernoulli process. The WF-IBP is then preferable in all settings where one is interested in a direct interpretation of the evolution of features. Second, the dIBP suffers from a less flexible boundary behaviour as it does not allow features to be born and die over time. Third, the dIBP is based on the stick-breaking construction of the IBP, which is only available for the one-parameter IBP; instead, our approach extends the two-parameter IBP and models the dimensionality of the feature allocation matrix and its sparsity independently.

2.4 Fixed-K approximation

In order to give more intuition on the exact inference with the WF-IBP that is developed in Section 2.5, we first describe a finite-dimensional approximation where the number of features is a finite number K, and show that this marginally converges to the WF-IBP as $K \to \infty$.

Assume that the random feature allocation matrix Z_t has a fixed number of features, say K. First, let $\alpha, \beta > 0$ and consider the beta-binomial model

$$\begin{aligned} Z_{ikt} \mid \{X_k(t) = x_k(t)\} &\overset{iid}{\sim} \text{Bernoulli}(x_k(t)), \\ X_k(t) &\overset{iid}{\sim} \text{Beta}\left(\frac{\alpha\beta}{K}, \beta\right), \end{aligned}$$

$\forall k = 1, \ldots, K, \forall i = 1, \ldots, N_t$. This coincides with the pre-limiting model of the two-parameter IBP presented in Ghahramani et al. [2007]. Then, for each feature, think of the $\text{Beta}\left(\frac{\alpha\beta}{K}, \beta\right)$ distribution as the stationary distribution of a W-F diffusion with parameters $\frac{\alpha\beta}{K} > 0$ and $\beta > 0$. This suggests making the model time-dependent by letting each feature evolve, starting at stationarity, as an independent W-F diffusion with these parameters. Generate for all times $t = t_0, \ldots, t_T$ the binary

variables z_{ikt} as

$$Z_{ikt} \mid \{X_k(t) = x_k(t)\} \overset{iid}{\sim} \text{Bernoulli}(x_k(t)),$$
$$X_k \sim \text{WF}\left(\frac{\alpha\beta}{K}, \beta\right),$$

$\forall k = 1, \ldots, K, \forall i = 1, \ldots, N_t$. In this way, the closer two time points, the stronger the dependency between the probabilities of a given feature (Figure 2.2). Moreover, as we assume the W-F diffusion to start at stationarity, this construction coincides marginally with the beta-binomial model. The parameters of the W-F diffusion are positive, so that neither fixation nor absorption ever occurs and the number K of features remains constant.

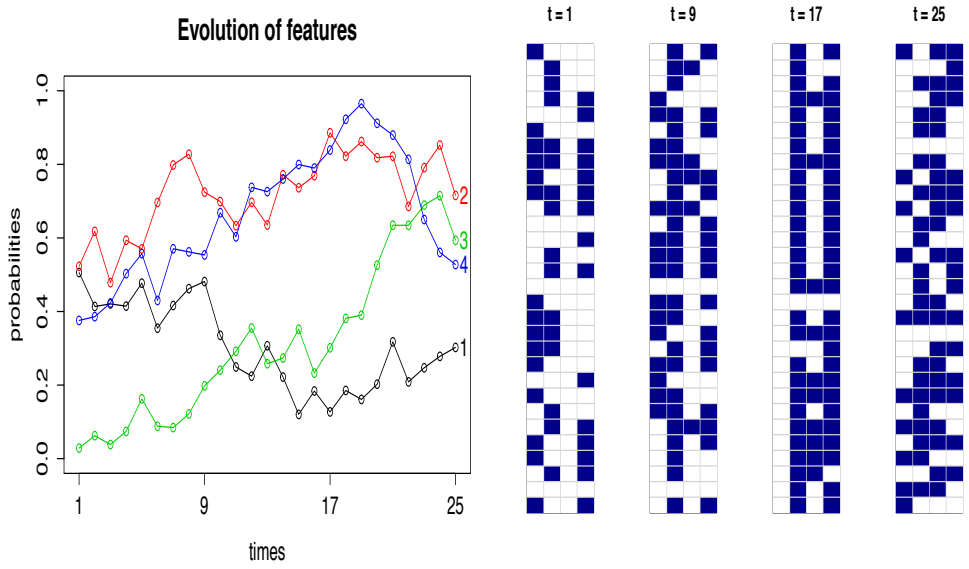


Figure 2.2: Left: underlying feature probabilities over time. Right: corresponding feature allocation matrices. Rows represent objects and columns features, which can be active (blue) or inactive (white).

2.4.1 Fixed-K MCMC inference

Given a set of observations D, a natural inference problem would be to recover the latent feature allocation matrices $Z = \{Z_t\}_{t=t_0}^{t_T}$ responsible for generating the observed data, the underlying feature probabilities X and their parameters ρ. Inference is straightforward; we propose the following updates.

- $Z \mid X, D, \rho$ via Gibbs sampling.
- $\rho \mid D, Z$ according to the likelihood model.
- $X \mid Z$ via Particle Gibbs.

Consider first the Gibbs sampling step to perform posterior inference over the matrices Z. Denote by $Z_{-(ik)t}$ all the components of the matrix Z_t excluding Z_{ikt}, and by Z_{i-kt} all the components in row i excluding k. We can easily derive for all t the distribution of a given component Z_{ikt} conditioning on the state of all other components $Z_{-(ik)t}$, on data D_{it}, on the parameters ρ and on the prior probability $X_k(t)$ of feature k. The full conditional probability of the entry Z_{ikt} being active is equal to

$$P(Z_{ikt} = 1 \mid Z_{-(ik)t}, X_k(t), D_{it}, \rho) \propto x_k(t) P(D_{it} \mid Z_{i-kt}, Z_{ikt} = 1, \rho). \tag{2.6}$$

By the same token, the full conditional probability of the entry Z_{ikt} being inactive is

$$P(Z_{ikt} = 0 \mid Z_{-(ik)t}, X_k(t), D_{it}, \rho) \propto (1 - x_k(t)) P(D_{it} \mid Z_{i-kt}, Z_{ikt} = 0, \rho). \tag{2.7}$$

As the matrices Z are conditionally independent given the feature probabilities X, equations (2.6) and (2.7) can be used to sample the matrices Z independently given the respective feature probabilities at each time. Note that the likelihood $P(D_{it} \mid Z_t, \rho)$ needs to be specified according to the problem at hand. A typical choice, detailed in Section 2.6, is the linear-Gaussian likelihood model, whose parameters can easily be integrated out [Griffiths and Ghahramani, 2011]. The update $\rho \mid D, Z$ over the feature parameters is also specific to the likelihood model and, as we will illustrate, can easily be derived in conjugate models such as the linear-Gaussian one.

Consider now the Particle Gibbs (PG) step [Andrieu et al., 2010] to perform Bayesian inference on the feature trajectories continuously over the interval $[t_0, t_T]$. As the prior probability of each feature is a Beta $\left(\frac{\alpha\beta}{K}, \beta\right)$ and the column-wise sums

of Z_{t_0} are realizations from binomial distributions, by conjugacy we have

$$X_k(t_0) \mid \{Z_{t_0} = z_{t_0}\} \sim \text{Beta}\left(\frac{\alpha\beta}{K} + n_{kt_0}, \beta + N_{t_0} - n_{kt_0}\right),$$

where $k = 1, \ldots, K$, and $n_{kt} := \sum_{i=1}^{N_t} z_{ikt}$ denotes the number of objects in matrix Z_t possessing feature k. This posterior distribution can be therefore used to draw the whole set of features at time t_0 and the trajectories in the interval $[t_0, t_T]$ can be obtained via PG.

More precisely, start with an initial reference trajectory $x^{r,k}_{t_0:t_T} := (x^{r,k}_{t_0}, \ldots, x^{r,k}_{t_T})$ for $k = 1, \ldots, K$ and, independently for each feature, iterate the following procedure. Draw a given number of particles from the posterior beta distribution at time t_0 and propagate them forward to time t_1 according to $\text{WF}(\frac{\alpha\beta}{K}, \beta)$. At time t_1, assign each of these particles and the reference feature a weight given by the binomial likelihood of seeing that feature active in $n(t)$ objects out of $N(t)$ in $Z(t)$, i.e., $x_k(t)^{n_{kt}}(1 - x_k(t))^{N_t - n_{kt}}$. Sample the weighted particles with replacement and propagate the off-springs forward. This corresponds to using a bootstrap filter with multinomial resampling, but other choices to improve on the performance of the sampler can be made [see Andrieu et al., 2010]. Repeat this procedure up to time t_T and sample only one particle at that time. Reject all the others and keep the trajectory that led to the sampled particle as the reference trajectory for the next iteration. Notice that the reference feature is kept intact throughout each iteration of the algorithm.

This procedure is illustrated more precisely by Algorithm 1, which needs to be iterated independently for each feature to provide posterior samples from their trajectories. To simplify the notation, we drop the index k and write $x^i_t | x^i_{t_0} \sim \text{WF}(\frac{\alpha\beta}{K}, \beta)$ for $t \in [t_0, t_1]$ to denote the following: simulate from a W-F diffusion with initial value $x(t_0)$ and set $X(t_1) = x(t_1)$, the value of the diffusion at time t_1.

2.4.2 Approximation for large K

As already noted, the marginal distribution with the fixed-K approximation corresponds to the beta-binomial model, which is the pre-limiting model of the two-parameter IBP. As a consequence, at any fixed time t and as $K \to \infty$, the fixed-K

Algorithm 1: Particle Gibbs

Input: Reference trajectory $x^r_{t_0:t_T}; M$.

Set $x^M_{t_0} = x^r_{t_0}$;

Draw $x^i_{t_0} \sim \text{Beta}(\frac{\alpha\beta}{K} + n_{t_0},\, \beta + N_{t_0} - n_{t_0})$ for $i = 1, \ldots, M-1$;

Simulate $x^i_t | x^i_{t_0} \sim \text{WF}(\frac{\alpha\beta}{K}, \beta)$ for $t \in [t_0, t_1]$ for $i = 1, \ldots, M-1$;

Set $x^M_{t_1} = x^r_{t_1}$;

Compute $w^i_{t_1} = (x^i_{t_1})^{n_{t_1}} (1 - x^i_{t_1})^{N_{t_1} - n_{t_1}}$ for $i = 1, \ldots, M$;

Sample $\bar{x}^i_{t_1}$ with $P(\bar{x}^i_{t_1} = x^i_{t_1}) \propto w^i_{t_1}$ for $i = 1, \ldots, M-1$;

Set $\bar{x}^M_{t_1} = x^r_{t_1}$;

Simulate $x^i_t | x^i_{t_1} \sim \text{WF}(\frac{\alpha\beta}{K}, \beta)$ for $t \in [t_1, t_2]$ for $i = 1, \ldots, M-1$;

Set $j \leftarrow 2$;

while $t_j < t_T$ **do**

  Set $x^M_{t_j} = x^r_{t_j}$;

  Compute $w^i_{t_j} = (x^i_{t_j})^{n_{t_j}} (1 - x^i_{t_j})^{N_{t_j} - n_{t_j}} w^i_{t_{j-1}}$ for $i = 1, \ldots, M$;

  Sample $\bar{x}^i_{t_j}$ with $P(\bar{x}^i_{t_j} = x^i_{t_j}) \propto w^i_{t_j}$ for $i = 1, \ldots, M-1$;

  Set $\bar{x}^M_{t_j} = x^r_{t_j}$;

  Simulate $x^i_t | \bar{x}^i_{t_j} \sim \text{WF}(\frac{\alpha\beta}{K}, \beta)$ for $t \in [t_j, t_{j+1}]$ for $i = 1, \ldots, M-1$;

  Set $j \leftarrow j + 1$;

end

Compute $w^i_{t_T} = (x^i_{t_T})^{n_{t_T}} (1 - x^i_{t_T})^{N_{t_T} - n_{t_T}} w^i_{t_{T-1}}$ for $i = 1, \ldots, M$;

Sample r_{new} with $P(r_{new} = i) \propto w^i_{t_T}$, where $i = 1, \ldots, M$;

Output: New reference trajectory $x^{r_{new}}_{t_0:t_T}$.

approximation converges to the two-parameter generalization of the IBP, which in turns coincides with the marginal distribution of the WF-IBP. An aspect of interest is then whether the whole dynamics of the fixed-K approximation can be used as a finite approximation of the infinite model in such a way that, the larger K, the better the approximation. Two caveats need to be noted. First, only in the infinite model can features be born. For large K, however, the number of particles in the fixed-K approximation whose mass is close to zero becomes so large that, with a sufficient amount of time, some of them gain enough mass to become 'visible'. The behavior of these particles resembles the behavior of the newborn features of the infinite model. Second, as the fixed-K approximation has an upwards drift equal to $\alpha\beta/K$, only the infinite model allows for features to be absorbed at 0. This discrepancy is however overcome by the fact that, when K goes to infinity, 0 behaves like an absorbing boundary, in that features get trapped at probabilities close to 0. For these reasons, a comparison of the two models requires relabeling the particles in the finite model in such a way that, whenever a particle goes below a certain threshold $\epsilon \approx 0$, it is considered to have gone extinct, while if its probability is below ϵ and later exceeds ϵ the particle is labeled as newborn. We choose this threshold to be $\epsilon = 1/K$, as then $\lim_{K\to\infty} \epsilon = 0$.

Taking these caveats into account, we performed an empirical comparison of the fixed-K approximation with the infinite model. Consider the joint distribution at two given time points $t_0 = 0$ and $t_1 = 1$ of the feature probabilities, first in the fixed-K and then in the infinite model. Separately for each model, we took 1000 samples of the feature probabilities at time 0 and at time 1, excluding the ones below $1/K$. Figure 2.3 shows the logarithm of these values for the two models, suggesting a remarkable similarity between the two underlying joint distributions. The validity of this comparison is supported by the maximum mean discrepancy (mmd) test [Gretton et al., 2006], which does not reject the null hypothesis of the two joint distributions being the same. Although this suggests a strong similarity between the dynamics of the fixed-K approximation and the infinite model, we leave a proof of the convergence of these joint distributions as $K \to \infty$ for future work.

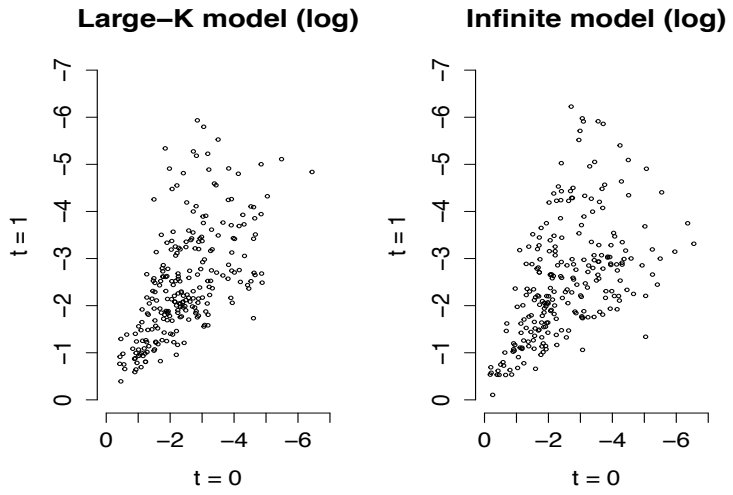


Figure 2.3: Scatterplot of the log feature probabilities greater than $1/K$ at time $t_0 = 0$ and at time $t_1 = 1$ to compare the fixed-K ($K = 1000$) and infinite model.

2.5 Exact MCMC inference

Building on the fixed-K approximation, we now develop a sophisticated MCMC algorithm for exact inference with the WF-IBP. The first point to notice is that, while in the fixed-K approximation the total number of features is constant over time and equal to K, in the WF-IBP model this is not a finite number. In order to use the WF-IBP for inference it is necessary to augment the state space with the features that are not seen in the feature allocation matrices, but simulating the dynamics of the PRF would require generating an infinite number of features, which is clearly unfeasible. One way to deal with this could be to resort to some sort of truncation, considering only features whose probability is greater than a given threshold and are likely to be seen in the data. We rather choose this truncation level adaptively by introducing a collection of slice variables $\{S_t\}_{t=1}^{T_t}$ and adopting conditional slice sampling [Walker, 2007; Teh et al., 2007]. This scheme, which is detailed in Section 2.5.1, has the advantage of making inference tractable without introducing approximations.

Partition the set of features into two subsets, one containing the features that have been seen at least once among times $t = t_0, \ldots, t_T$, and the other containing the features that have never been seen for all $t = t_0, \ldots, t_T$, so that $X = X_{\text{seen}} \cup X_{\text{unseen}}$.

Since the unseen features cannot be identified individually based on the matrices Z, and as all features are conditionally independent given Z, we consider seen and unseen features separately. As for the seen features, we use the same Particle Gibbs scheme as in the fixed-K approximation. As for the unseen features, we simulate them via a thinning scheme. The exact MCMC inference scheme can be then summarized by the following updates.

- $Z \mid X, S, D, \rho$ via Gibbs sampling.
- $S \mid Z, X$ via slice sampling.
- $\rho \mid D, Z$ according to the likelihood model.
- $X_{\text{seen}} \mid Z$ via Particle Gibbs.
- $X_{\text{unseen}} \mid S$ via thinning.

We present each of these steps in Sections 2.5.1 and 2.5.2, which build on the simpler inference scheme developed in Section 2.4.

2.5.1 Gibbs and slice sampling

The first step is to augment the parameter space with a set of slice variables. Given the feature allocation matrices Z_t at times $t = t_0, \ldots, t_T$, draw a slice variable $S_t \sim \text{Uniform}[0, x^{min}(Z_t)]$ for each time t, where $x^{min}(Z_t)$ is the minimum among the probabilities of the features seen in the feature matrix Z_t. In this way, when conditioning on the value s_t of the slice variable, we have a truncation level s_t and only need to sample the finite number of features whose probability is above this threshold [Teh et al., 2007]. In other words, for all $t = t_0, \ldots, t_T$, we only need to update the columns of Z_t whose corresponding feature probability $x_k(t)$ is greater than or equal to the slice variable s_t (note that these include both seen and currently unseen features). Observe that we defined a different slice variable for each time point, while an alternative choice could have been drawing a single slice from a uniform between 0 and the minimum feature probability across all times. Having multiple slice variables makes it possible to simulate fewer feature trajectories while keeping inference exact, reducing the computational cost of simulating features with

small probabilities of being active. Although this comes at the cost of a larger number of parameters, in experiments we find that having multiple slices does not compromise mixing nor predictive performance.

Accounting for the slice variables, the full conditional probability of the entry Z_{ikt} being active is directly proportional to

$$\begin{aligned} &P(Z_{ikt} = 1 \mid Z_{-(ik)t}, X_k(t), D_{it}, \rho) P(S_t \mid Z_{ikt} = 1, Z_{-(ik)t}) \propto \\ &\quad x_k(t) P(D_{it} \mid Z_{i-kt}, Z_{ikt} = 1, \rho) \frac{1[0 \leq s_t \leq x^{min}(Z_{-(ik)t}, Z_{ikt} = 1)]}{x^{min}(Z_{-(ik)t}, Z_{ikt} = 1)}, \end{aligned} \tag{2.8}$$

while the full conditional probability of the entry Z_{ikt} being inactive is directly proportional to

$$\begin{aligned} &P(Z_{ikt} = 0 \mid Z_{-(ik)t}, X_k(t), D_{it}, \rho) P(S_t \mid Z_{ikt} = 0, Z_{-(ik)t}) \propto \\ &\quad (1 - x_k(t)) P(D_{it} \mid Z_{i-kt}, Z_{ikt} = 0, \rho) \frac{1[0 \leq s_t \leq x^{min}(Z_{-(ik)t}, Z_{ikt} = 0)]}{x^{min}(Z_{-(ik)t}, Z_{ikt} = 0)}. \end{aligned} \tag{2.9}$$

The term $P(S_t \mid Z_t)$ is not constant as updating Z_{ikt} for a currently unseen feature may modify the value of the minimum probability of the active features.

2.5.2 Particle Gibbs and thinning

Assume that the feature allocation matrices Z_t are given at the time points $t = t_0, \ldots, t_T$ and we are interested in inferring the probabilities X of the underlying features. This section gives the details of inference for each of the following subpartitions of seen and unseen features: features seen for the first time at a given time t_j (for $j = 0, \ldots, T$), unseen features alive at time t_0 and unseen features born between any two consecutive times t_j and t_{j+1} (for $j = 0, \ldots, T-1$).

Seen features

As already mentioned, we can apply PG to sample from the posterior trajectories of the seen features. In particular, for features that are seen at time t_0 we can simply apply Algorithm 1 as in the fixed-K approximation by replacing the term $\frac{\alpha\beta}{K}$ with 0. This is possible as observing a feature allocation matrix Z_t updates the prior probability of features as in the posterior beta process [Thibaux and Jordan, 2007],

meaning that we can draw each seen feature k from a Beta(n_{kt}, $\beta + N_t - n_{kt}$) (recall that n_{kt} is the number of objects in which feature k is active at time t).

More generally, consider features that are seen for the first time at a given time t_j. As they cannot be identified individually based on any feature matrix Z_{t_k} for $k < j$, these features need to be drawn from the posterior beta process at time t_j and propagated both forward and backwards. Note that simulating from the W-F diffusion backwards in time is not a problem as each W-F$(0, \beta)$ diffusion is time-reversible with respect to the speed density of the PRF [Griffiths, 2003]. The additional backward propagation requires adjusting Algorithm 1, already modified by replacing $\frac{\alpha\beta}{K}$ with 0, by further replacing the steps before the while loop with Algorithm 2, where for simplicity we describe the particular case of features seen for the first time at time t_1. This description can be easily generalized to features that are seen for the first time at a generic time point $t \in \{t_0, \ldots, t_T\}$.

Algorithm 2: PG: features seen for the first time at time t_1.

Input: Reference trajectory $x^r_{t_0:t_T}$; M.
Set $x^M_{t_1} = x^r_{t_1}$;
Draw $x^i_{t_1} \sim \text{Beta}(n_{t_1},\, \beta + N_{t_1} - n_{t_1})$ for $i = 1, \ldots, M-1$;
Set $x^M_{t_0} = x^r_{t_0}$;
Simulate $x^i_t | x^i_{t_1} \sim \text{WF}(0, \beta)$ backwards for $t \in [t_1, t_0]$ for $i = 1, \ldots, M-1$;
Set $x^M_{t_2} = x^r_{t_2}$;
Simulate $x^i_t | x^i_{t_1} \sim \text{WF}(0, \beta)$ for $t \in [t_1, t_2]$ for $i = 1, \ldots, M-1$;
Compute $w^i_{t_0} = (1 - x^i_{t_0})^{N_{t_0}}$ for $i = 1, \ldots, M$;
Compute $w^i_{t_2} = (x^i_{t_2})^{n_{t_2}} (1 - x^i_{t_2})^{N_{t_2} - n_{t_2}} w^i_{t_0}$ for $i = 1, \ldots, M$;
Draw $\bar{x}^i_{t_2}$ with $P(\bar{x}^i_{t_2} = x^i_{t_2}) \propto w^i_{t_2}$ for $i = 1, \ldots, M-1$;
Set $\bar{x}^M_{t_2} = x^r_{t_2}$;
Simulate $x^i_t | x^i_{t_2} \sim \text{WF}(0, \beta)$ for $t \in [t_2, t_3]$ for $i = 1, \ldots, M-1$;
Set $j \leftarrow 3$;

Unseen features

We now describe a thinning scheme to simulate the unseen features alive at time t_0. Denote the slice variable values at each time by $s_{t_0}, \ldots, s_{t_T}$ and note that sampling the set of unseen features from the truncated posterior beta process at time t means drawing samples from a Poisson process on $[s_t, 1)$ with rate measure $x^{-1}(1-x)^{\beta + N_t - 1} dx$ [Thibaux and Jordan, 2007], which yields only a finite number of

features whose probability is larger than s_t. First, draw the unseen features from the truncated posterior beta process at time t_0. Then, propagate them forward to time t_1 according to the W-F diffusion and accept them with probability $(1-x(t_1))^{N(t_1)}$, namely the binomial likelihood of not seeing them in any object at time t_1. Finally, iterate this propagation and rejection steps up to time t_T. The details of this thinning scheme are given in Algorithm 3.

Algorithm 3: Thinning: unseen features alive at time t_0

Draw from a Poisson process on $[s_{t_0}, 1)$ with rate measure $\alpha x^{-1}(1-x)^{\beta+N_{t_0}-1}dx$ and denote by $\{x^i_{t_0}\}_{i\in A}$ the resulting candidate particles;
Set $j \leftarrow 1$;
while $t_j < t_T$ **do**
  Simulate $x^i_t | x^i_{t_j} \sim \mathrm{WF}(0,\beta)$ for $t \in [t_j, t_{j+1}]$ for all $i \in A$;
  Accept $x^i_{t_{j+1}}$ with probability $(1-x^i_{t_{j+1}})^{N_{t_{j+1}}}$ for all $i \in A$;
  Remove from A the indices of the rejected particles;
  Set $j \leftarrow j+1$;
end
Output: Trajectories $\{x^i_{t_0:t_T}\}_{i\in A}$ of the unseen features alive at time t_0 from the truncated $\mathrm{PRF}(\alpha,\beta)$.

Notice that simulating the trajectories of the unseen features born between time t_0 and t_1 is equivalent to Algorithm 3 from time t_1 onwards. The only difference is that these features, drawn at time t_1, need to be simulated backwards to time t_0 as well, hence the additional backward simulation followed by the rejection step in the for loop of Algorithm 4. If a feature that is simulated backwards from time t_1 to t_0 has probability 0 by time t_0, then it is a newborn feature and is accepted with probability 1. On the other hand, if its probability at time t_0 is between 0 and s_{t_0}, the particle belongs to the category of features that were alive and unseen at time t_0. Accepting them with probability $(1-x(t_0))^{N_{t_0}}$ compensates for the features that were below the truncation level s_{t_0} in Algorithm 3 and were thus not simulated at time t_0. In this way, only the features whose mass is below the slice variables s_t at all times $t \in \{t_0, \ldots, t_T\}$ are not simulated. The exactness of the overall MCMC scheme is preserved by the fact that those features are inactive in all the feature allocation matrices by the definition of slice variable.

Finally note that, for simplicity's sake, Algorithm 4 describes only how to

Algorithm 4: Thinning: unseen features born between time t_0 and t_1

Draw from a Poisson process on $[s_{t_1}, 1)$ with rate measure $\alpha x^{-1}(1-x)^{\beta+N_{t_1}-1}dx$ and denote by $\{x^i_{t_1}\}_{i\in A}$ the resulting candidate particles;
Simulate $x^i_t|x^i_{t_0} \sim \text{WF}(0,\beta)$ for $t \in [t_0, t_1]$ for all $i \in A$;
for *all* $i \in A$ **do**
  if $x^i_{t_0} > s_{t_0}$ **then**
    Reject $x^i_{t_0}$;
    Set $A \leftarrow A \setminus \{i\}$;
  else
    Accept $x^i_{t_0}$ with probability $(1-x^i_{t_0})^{N_{t_0}}$;
  end
end
Set $j \leftarrow 1$;
while $t_j < t_T$ **do**
  Simulate $x^i_t|x^i_{t_j} \sim \text{WF}(0,\beta)$ for $t \in [t_j, t_{j+1}]$ for all $i \in A$;
  Accept $x^i_{t_{j+1}}$ with probability $(1-x^i_{t_{j+1}})^{N_{t_{j+1}}}$ for all $i \in A$;
  Remove from A the indices of the rejected particles;
  Set $j \leftarrow j+1$;
end
Output: Trajectories $\{x^i_{t_0:t_T}\}_{i\in A}$ of the unseen features born between time t_0 and t_1 from the truncated PRF(α,β).

simulate the unseen features that were born between times t_0 and t_1, but the procedure needs to be generalized to account for the features born between any two consecutive time points t_j and t_{j+1}, where $j = 0, \ldots, T-1$. In order to do this, it is sufficient to draw the candidate particles at every time t_{j+1}, with $j = 0, \ldots, T-1$, propagate them backwards until time t_0 and thin them as follows: if their mass exceeds s_t at any $t \in \{t_0, \ldots, t_T\}$, then they are rejected; otherwise, at each backward propagation to time $t \in \{t_0, \ldots, t_{T-1}\}$ they are accepted with probability $(1-x(t))^{N_t}$.

2.6 Application: linear-Gaussian likelihood model

The WF-IBP we have described defines a prior over latent feature allocation matrices Z and the corresponding feature probabilities X exhibiting a dependency structure over time. The next step is to relate Z to the observed data by means of a given likelihood model. The first model we explore is the linear-Gaussian

likelihood, a very common choice in latent feature models [e.g., Doshi-Velez and Ghahramani, 2009; Griffiths and Ghahramani, 2011; Gershman et al., 2015].

Assume that the collection of observations O_t at time $t = t_0, \ldots, t_T$ is in the form of an $N \times D$ matrix generated by the matrix product $O_t = Z_t \times A + \epsilon_t$. Z_t is the $N \times K$ binary matrix of feature assignments at time t and A is a $K \times D$ factor matrix whose rows represent the feature parameters ρ. The matrix product is the way Z_t determines which features are active in each observation, and ϵ_t is a $N \times D$ Gaussian noise matrix, whose entries are assumed to be distributed as independent $\mathcal{N}(0, \sigma_X^2)$. A typical inference problem is to infer both the feature allocation matrices Z and the factor matrix A. In order to achieve this, we place on each element of A an independent prior $\mathcal{N}(0, \sigma_A^2)$ and on the hyper-parameter σ_A^2 an inverse-gamma prior $\Gamma^{-1}(1, 1)$. This choice of priors is convenient as it is easy to obtain the posterior distributions of σ_A^2 and A [for the case $T = 1$, see Doshi-Velez and Ghahramani, 2009].

For simplicity of notation, consider a fixed number of features K. Denote by $\bar{Z}$ the $TN \times K$ matrix obtained by concatenating the feature matrices Z vertically, and by $\bar{O}$ the $TN \times D$ matrix obtained by combining the observations $\{O_t\}_{t=t_0}^{t_T}$ in the same way. The posterior of A is matrix Gaussian with the following mean μ^A (a $K \times D$ matrix) and, for each column of A, the following covariance matrix Σ^A (a $K \times K$ matrix).

$$\mu^A = \left(\bar{Z}^T \bar{Z} + \frac{\sigma_X^2}{\sigma_A^2} I \right)^{-1} \bar{Z}^T \bar{O}$$
$$\Sigma^A = \sigma_X^2 \left(\bar{Z}^T \bar{Z} + \frac{\sigma_X^2}{\sigma_A^2} I \right)^{-1}.$$

By conjugacy, the posterior distribution for σ_A^2 is still inverse gamma with updated parameters, namely

$$\sigma_A^2 \sim \Gamma^{-1} \left(1 + \frac{1}{2} KD, 1 + \frac{1}{2} \sum_k \sum_d A_{kd}^2 \right).$$

2.6.1 Simulations and results

We tested the WF-IBP combined with a linear-Gaussian likelihood on a variety of synthetic data sets. Starting from the fixed-K approximation, we generated $N = 50$ observations at each of 40 equally-spaced time points as in the linear-Gaussian model. The true factor matrix A contained $K = 3$ latent features in the form of binary vectors of length $D = 30$. Their probability of being active was determined continuously over time by three independent W-F$(1, 1)$ diffusions, simulated for 0.01 diffusion time-units between every two consecutive time points. The resulting observations were corrupted by a large amount of noise ($\sigma_X = 0.5$). 1000 iterations of the overall algorithm were performed, choosing a burn-in period of 100 iterations and setting the time-units and drift parameters of the W-F diffusion equal to the true ones in the PG update. As ground truth was available, we were able to test the ability of the algorithm to recover the true feature allocation matrices, the latent feature parameters and their probabilities over time.

Figure 2.4-top-left compares the true underlying feature matrices at times $t = \{1, 14, 27, 40\}$ in terms of the most frequently active features in the posterior mean matrices, where a feature is set to be active if that is the case in more than half of the samples of the Markov chain. The resulting mean matrices almost perfectly match the true underlying feature matrices. Figure 2.4-top-right compares the trajectories over time of the true feature probabilities with the inferred ones. The latter tend to be less than two standard deviations away from the former, meaning that the true feature trajectories are closely tracked. Figure 2.4-bottom-left compares the three features represented by the true factor matrix A and the ones in the posterior mean matrix $\hat{A}$, showing that the algorithm was able to recover accurately the hidden features underlying the noisy observations. Figure 2.4-bottom-right plots the log-likelihood at each iteration, showing that the algorithm converged quickly, namely in fewer than 50 iterations.

Then, we tested the ability of the slice sampler-based algorithm to recover the correct number of latent features when given a similar set of synthetic data, this time consisting of 4 latent features evolving over 6 time points. The algorithm was initialized with one feature and run for 3300 iterations with a burn-in period of 1000

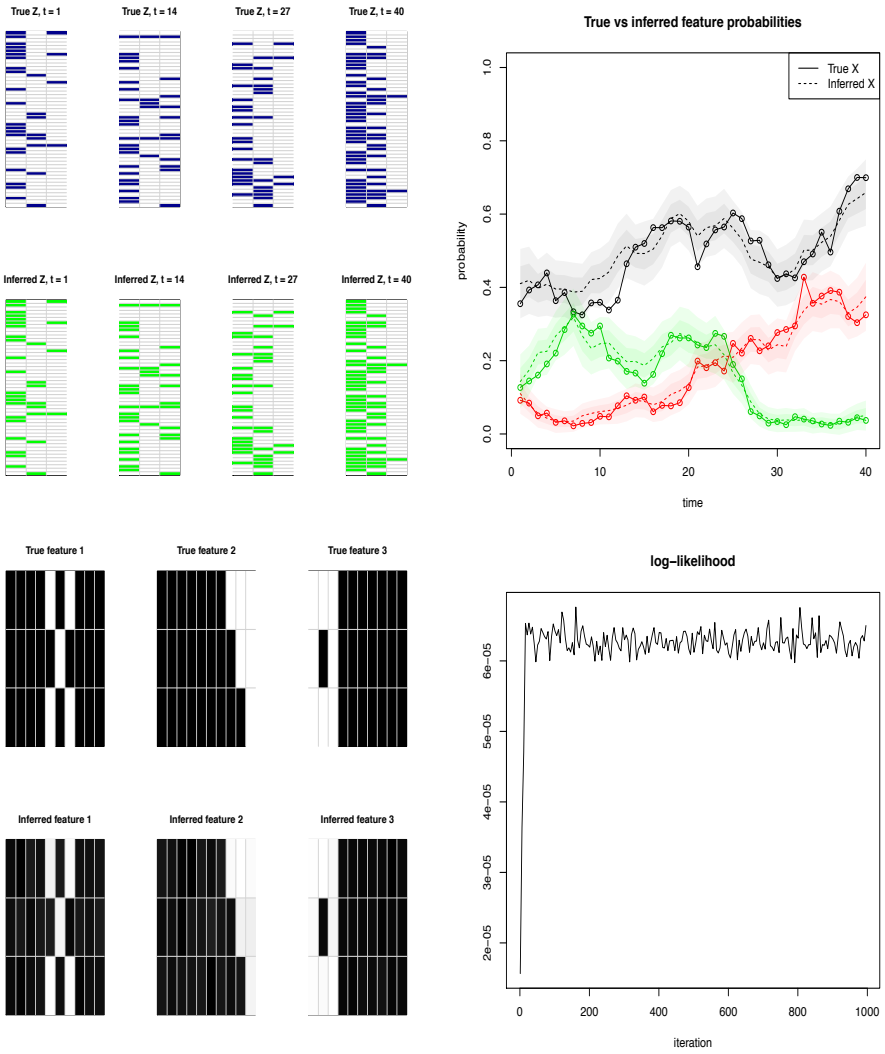


Figure 2.4: Fixed-K approximation. Top-Left: Subset of true feature allocation matrices vs inferred ones. Top-Right: True vs inferred feature probabilities over time (the dark and the light shaded areas respectively indicate one and two standard deviations about the posterior mean). Bottom-Left: True vs inferred features (black and white entries respectively correspond to 0 and 1, while the shades of grey to the values in between). Bottom-Right: Convergence of the log-likelihood of the observed data.

iterations. As in the finite case, the true underlying feature allocation matrices and feature probabilities were closely recovered as illustrated by the top row of Figure 2.5. The bottom row of Figure 2.5 shows that the features were reconstructed accurately and their correct number detected in about 700 iterations.

2.6.2 Model misspecification

In several applications, ranging from topic to video modeling, time dependence is an evident phenomenon and thus it is straightforward to assess whether the WF-IBP is a suitable model choice. In contrast, in settings where forms of time-dependency are not evident, we recommend using the standard IBP. In this section, we focused on the ability of Particle Gibbs to track the feature trajectories in a wider variety of settings and under model misspecification. We generated a set of feature allocation matrices obtained by letting three feature probabilities evolve over time, first by simulating standard W-F diffusions and then by introducing jumps and spikes. Figure 2.6 confirms the robustness of the algorithm in the presence of mismatches between the W-F diffusion and the true process determining the feature trajectories. We simulated a set of 20 feature allocation matrices under an increasing amount of mismatch, first by introducing a jump of increasing size from time 10 to 11, and then by adding a spike of increasing sharpness at time 10. Figure 2.6-left shows that, although larger jump sizes lead to larger discrepancies between the true and inferred trajectories around the jump, the former are always less than two standard deviations away from the latter. Figure 2.6-right shows that in all three cases the algorithm is able to detect the presence of the spike at time 10. Figure 2.7 illustrates the performance of the algorithm on a decreasing number of observations. The results show that the posterior mean of the target distribution closely corresponds to the true feature probabilities and, given a sufficient number of observations, always fall within the interval given by two standard deviations about the posterior mean.

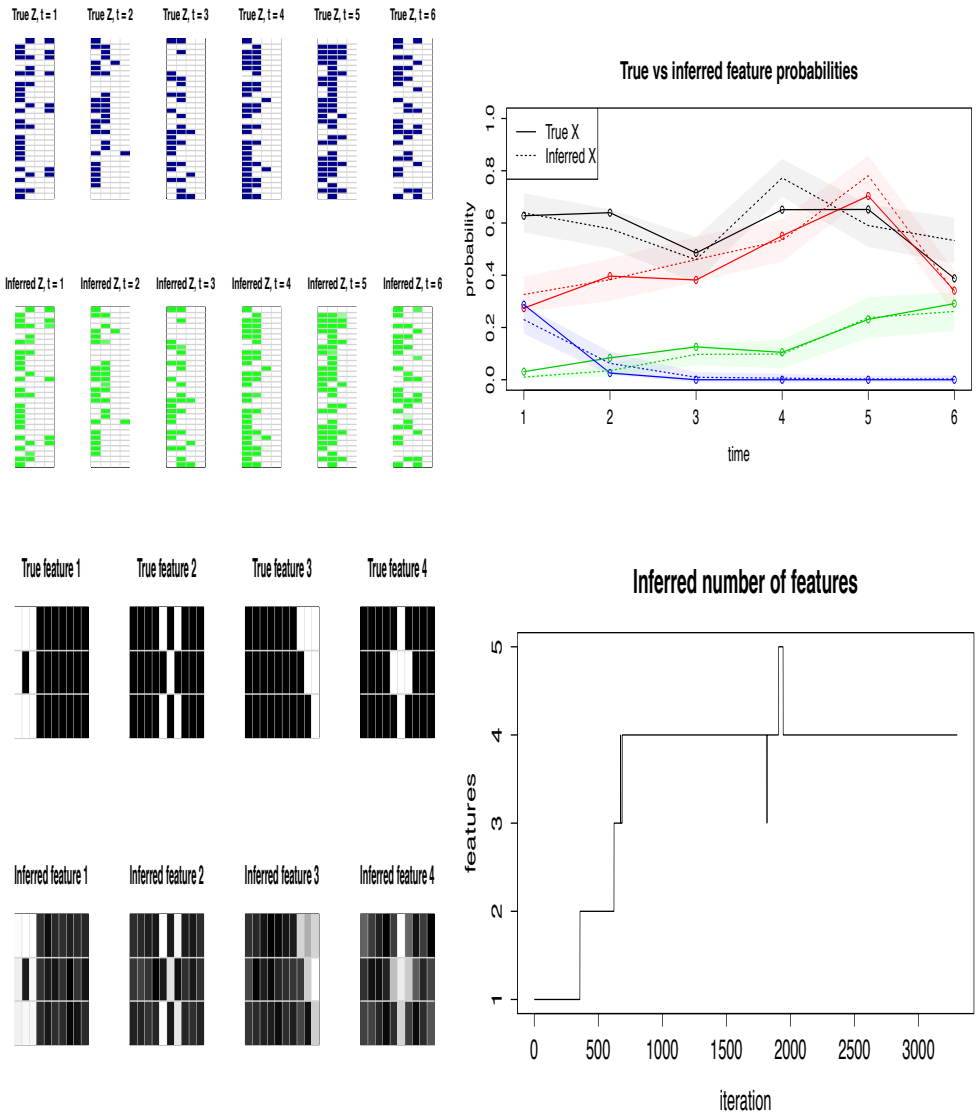


Figure 2.5: WF-IBP. Top-Left: True vs inferred feature allocation matrices. Top-Right: True vs posterior mean feature trajectories (the shaded areas represent one standard deviation). Bottom-Left: Comparison between true and inferred features. Bottom-Right: Convergence to the true number of features.

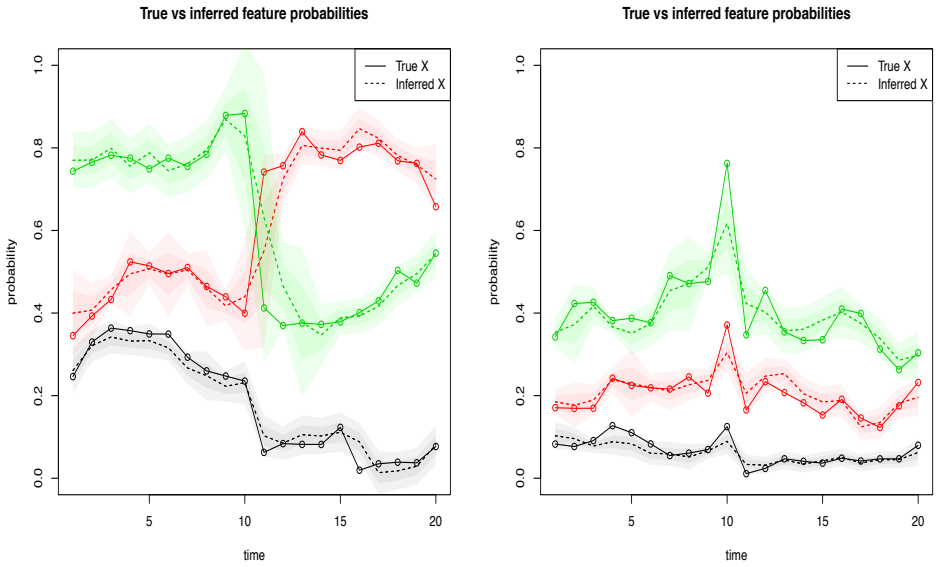


Figure 2.6: Comparison between the posterior means and the true feature probabilities under an increasing amount of model misspecification. The dark and light shaded areas respectively correspond to 1 and 2 standard deviations about the posterior mean. Left: Jump of increasing size between times 10 and 11. Right: Spike of increasing sharpness at time 10.

2.7 Topic modeling application

In this section we apply the WF-IBP to the modeling of corpora of time-stamped text documents. This is a natural application as documents can be seen as arising from an unknown number of latent topics whose popularity is evolving over time. A related model to achieve this goal is described in Blei and Lafferty [2006], where a Gaussian state space model captures the evolution of topics in such a way that both the content of topics and their proportions evolve over time. This work has had a great impact in the topic modeling community, and anticipates a number of directions for future work that we address here. Specifically their model is parametric, in that the number of topics K is fixed and needs to be pre-specified. The authors claim that it would desirable to drop this assumption to have a more flexible model and, in particular, foresee a process involving births and deaths of topics. The WF-IBP topic model elegantly achieves these goals. Unlike Blei and Lafferty [2006] we focus on the evolution of topic probabilities rather than topic contents, noting that the modeling of time-varying topic contents is orthogonal to our work

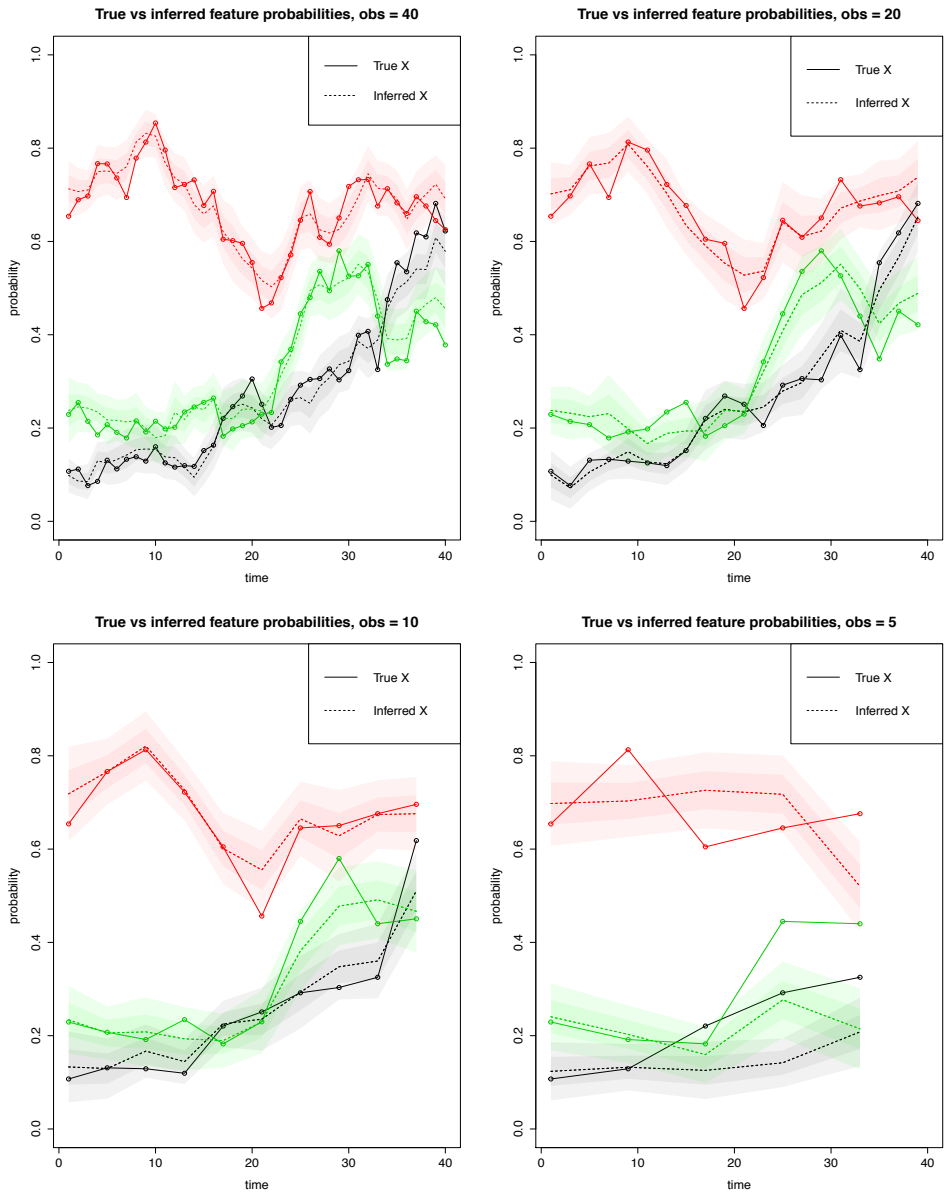


Figure 2.7: Comparison between true feature probabilities and posterior means obtained via Particle Gibbs for a varying number of observations. The dark and light shaded areas respectively correspond to 1 and 2 standard deviations about the posterior mean.

and could be incorporated into the WF-IBP in future developments. Another class of models called Dirichlet processes aim at modeling the evolution of topics in a time-dependent and nonparametric way. Some of these models, however, assume the evolution of topic probabilities to be unimodal [e.g., Rao and Teh, 2009], while others are HDP-based [Ahmed and Xing, 2012; Dubey et al., 2013] and implicitly assume a positive correlation between the probability of a topic being active and the proportion of that topic within each document. Coupling topic proportions and topic probabilities is undesirable as rare topics may account for a large proportion of words in the few documents in which they appear. Our nonparametric topic model decouples the probability of a topic and its proportion within documents and offers a flexible way to model topic evolutions over time. We achieve this by incorporating time-dependency into the focused topic model presented in Williamson et al. [2010b], which makes use of the IBP to select the finite number of topics that each document treats.

2.7.1 The WF-IBP topic model

First consider the case in which the number of topics K underlying the corpus of seen documents is known. Define topics as probability distributions over a dictionary of D words and model them as $(\rho_k)_{k=1}^K \overset{iid}{\sim} \text{Dirichlet}(\bar{\eta})$, given a vector $\bar{\eta}$ of length D. Let ρ be the resulting vector and assume the components of $\bar{\eta}$ to be all equal to a constant $\eta > 0$. Consider the usual setting in which the time-dependent popularity of topic (feature) k is denoted by X_k and the binary variables Z_{ikt} indicate whether document i contains topic k at time t. Then, for all $t = t_0, \ldots, t_T$ and $k = 1, \ldots, K$, sample

$$\begin{aligned}
\theta_{it} \mid \{Z_{it} = z_{it}, \phi_t = \phi'_t\} &\sim \text{Dirichlet}(z_{it} \circ \phi'_t), \quad \forall i = 1, \ldots, N_t, \\
\phi_{kt} &\sim \text{Gamma}(\gamma, 1), \\
(Z_{ikt})_{i=1}^{N_t} \mid \{X_k = x_k(t)\} &\overset{iid}{\sim} \text{Bernoulli}(x_k(t)), \\
X_k &\sim \text{WF}\left(\frac{\alpha\beta}{K}, \beta\right),
\end{aligned}$$

where ϕ_{kt} is the kth component of ϕ_t, a K-long vector of topic proportions, and θ_{it} the ith row of θ_t, a $N_t \times K$ matrix with the distributions over topics for each document at time t. The operation $z_{it} \circ \phi_t$ stands for the Hadamard product between z_{it} and ϕ_t and the Dirichlet is defined over the positive components of the resulting vector. While the topic allocation matrix Z_t encodes which subset of the K topics appears in each document at time t, the variables ϕ_{kt} are related to the proportion of words that topic k explains within each document. Unlike HDP-based models, these two quantities are here modeled independently.

For every document $i = 1, \ldots, N_t$, draw the total number of words from a negative-binomial $W_{it} \sim \mathrm{NB}(\sum_k z_{ikt}\phi_{kt}, 1/2)$ and, for each word w_{ilt}, $l = 1, \ldots, W_{it}$, sample first the topic assignment

$$a_{ilt} \mid \{\theta_{it} = \theta'_{it}\} \sim \mathrm{Categorical}(\theta'_{it})$$

and then the word

$$w_{ilt} \mid \{a_{ilt} = a'_{ilt}, \rho = \rho'\} \sim \mathrm{Categorical}(\rho'_{a'_{ilt}}).$$

Assume now that the number of potential topics K needs to be learned from the data. The nonparametric extension of this model is easily obtained by replacing the process generating the topic allocation matrices with the WF-IBP, so that topics arise as in the PRF and evolve as independent $\mathrm{WF}(0, \beta)$. The feature allocation matrices can be drawn as described in Section 2.9.2. In this way, we obtain a time-dependent extension of the IBP compound Dirichlet process presented in Williamson et al. [2010b].

2.7.2 Posterior inference

In order to infer the latent variables of the model, it is convenient to integrate out the parameters ρ and θ. This can be done easily thanks to the conjugacy between the Dirichlet and the Categorical distribution. In this way, we can run a Gibbs sampler for posterior inference only over the remaining latent variables and are able to follow the derivation of conditionals given by Williamson et al. [2010a].

Note that, in our case, we have introduced the slice variable and do not integrate out the topic allocation matrix. Denote by W the complete set of words and by A the complete set of topic assignments a_{ilt} for all times $t = t_0, \ldots, t_T$, documents $i = 1, \ldots, N_t$ and words $l = 1, \ldots, W_{it}$. Denote by S_t the slice variable and by W_t the complete set of words at time t. The conditional distributions that we need to sample from for all times $t = t_0, \ldots, t_T$ are

$$
\begin{aligned}
&p(A_t \mid Z_t, w, \phi_t),\\
&p(\phi_t, \gamma \mid A_t, W_t, Z_t),\\
&p(S_t \mid Z_t, X(t)),\\
&p(Z_t \mid A_t, X(t), \phi_t, S_t).
\end{aligned}
$$

Conditioning on all the other topic assignments a_{-il}, each topic assignment a_{ilt} can be sampled from

$$
p(a_{ilt} = k \mid a_{-il}, Z_t, W, \phi_t) \propto (n_k^{w_{il}} + \eta)\frac{n_{kt}^i + \phi_{kt} z_{ikt}}{n_k + \eta D - 1},
$$

where $n_k^{w_{il}}$ denotes the number of times that word w_{il} has been assigned to topic k excluding assignment a_{il}, n_{kt}^i the number of words assigned to topic k in document i excluding assignment a_{il} and n_k the total number of words assigned to topic k.

After placing a hyper-prior on $p(\gamma)$, we can sample ϕ and γ via a Metropolis-Hastings step. Indeed, we know that

$$
p(\phi_{kt}, \gamma \mid A, Z_t) \propto \frac{\phi_{kt}^{\gamma-1} e^{-\phi_{kt}}}{\Gamma(\gamma)} p(\gamma) \prod_{i=1}^{N_t} \frac{\Gamma(\phi_{kt} z_{ikt} + n_{kt}^i)}{\Gamma(\phi_{kt} z_{ikt}) n_{kt}^i! 2^{\phi_{kt} z_{ikt} + n_{kt}^i}}.
$$

Conditioning on Z_t, the slice variable is sampled according to its definition:

$$
p(S_t \mid Z_t, X(t)) = \frac{1}{x^{min}(t)} 1_{s_t}([0, x^{min}(t)]),
$$

where $x^{min}(t)$ is the minimum among the probabilities of the active topics at time t. As for the feature allocation matrices Z, we sample only the finite number of its components whose topic probability $x_k(t)$ is greater than the slice variable S_t.

Assume we are sampling each entry Z_{ikt} sequentially and denote respectively by $x_1^{min}(t)$ and $x_0^{min}(t)$ the minimum active topic probability in the cases $Z_{ikt} = 1$ and $Z_{ikt} = 0$. Let n_{ikt} denote the total number of words assigned to topic k in document i at time t. Then we have that

$$p(Z_{ikt} = 1 \mid A, x_k(t), \phi_{kt}) = \begin{cases} 1, & \text{if } n_{ikt} > 0 \\ \frac{x_k(t)x_0^{min}(t)}{x_k(t)x_0^{min}(t)+2^{\phi_{kt}}(1-x_k(t))x_1^{min}(t)}, & \text{if } n_{ikt} = 0. \end{cases}$$

$$p(Z_{ikt} = 0 \mid A, x_k(t), \phi_{kt}) = \begin{cases} 0, & \text{if } n_{ikt} > 0 \\ \frac{2^{\phi_{kt}}(1-x_k(t))x_1^{min}(t)}{x_k(t)x_0^{min}(t)+2^{\phi_{kt}}(1-x_k(t))x_1^{min}(t)}, & \text{if } n_{ikt} = 0. \end{cases}$$

The full conditional distributions presented so far are derived in Appendix C. As in the linear-Gaussian case, when considering the probability of setting $Z_{ikt} = 1$ for a feature that is currently inactive across all observations at time t, it is necessary to jointly propose a new value ϕ_{kt} by drawing it from its prior distribution Gamma$(\gamma, 1)$. Finally, inference on the trajectories of the topic probabilities is a direct application of the Particle Gibbs and thinning scheme outlined for the general case.

2.7.3 Topic model: simulation and results

At each of 4 time points, a small corpus of $N = 30$ documents was simulated by selecting up to $K = 4$ latent topics for each document and picking words from a dictionary of $D = 100$ words. The hyper-parameter of the Dirichlet prior over words was chosen to be a vector with components equal to $\eta = 0.1$, a Gamma(5,1) hyper-prior was placed on γ and we let topic probabilities evolve as independent W-F$(1, 1)$ diffusions with 0.1 diffusion time-units between each observation. Assume K is known and focus on inference over the remaining parameters. Fix the time-units and drift parameters of the W-F diffusion to their true values in the PG update. We ran the Gibbs sampler for 3000 iterations with a burn-in period of 300 iterations. The log-likelihood converged in about 500 iterations (Figure 2.8) and the algorithm was able to infer closely the latent topic allocation matrices (Figure

2.9-left). The percentage of words assigned to the correct topics were 81% at t_1, 82% at t_2, 83% at t_3 and 85% at t_4.

A Monte Carlo estimate for the probability of word $w = 1, \ldots, D$ under topic k is

$$\hat{\rho}_{kw} = \frac{n_k^w + \eta}{n_k + D\eta},$$

where n_k^w is the number of times word w has been assigned to topic k, n_k is the total number of words assigned to topic k and D is the number of words in the dictionary. A Monte Carlo estimate for the probability of topic k in document i is

$$\hat{\theta}_{ikt} = \frac{n_{ikt} + z_{ikt}\phi_{kt}}{\sum_k (n_{ikt} + z_{ikt}\phi_{kt})}. \tag{2.10}$$

These two quantities are given by the posterior mean of the Dirichlet distribution under a categorical likelihood. $\hat{\rho}_{kw}$ has been used to plot the posterior distribution over words in Figure 2.9-right. These results confirm the ability of the algorithm to recover ground truth and provide useful information both at word and topic level.

Finally, we tested the ability of the algorithm to reconstruct topics when presented with a decreasing number of observed documents. 100 documents were simulated at each of two time points as described above. Figure 2.10 shows how well the probability of the 10 most likely words of the first topic was reconstructed for $N = 60$, 40, 20 and 10 observed documents per time point. As expected, the more documents are observed the more accurate the reconstruction of topics is, with a drop in performance when only 10 documents per time point are observed.

Comparison: static model and hierarchical model

We now demonstrate the advantages of modeling time-dependency by comparing our model with two alternative versions. First, we consider a static counterpart where time is not modeled and thus information about the time-stamp of documents is not exploited. Second, we consider a hierarchical model where feature probabilities at each time point are distributed as conditionally independent beta processes given a lower-layer beta process [Thibaux and Jordan, 2007]; note that in

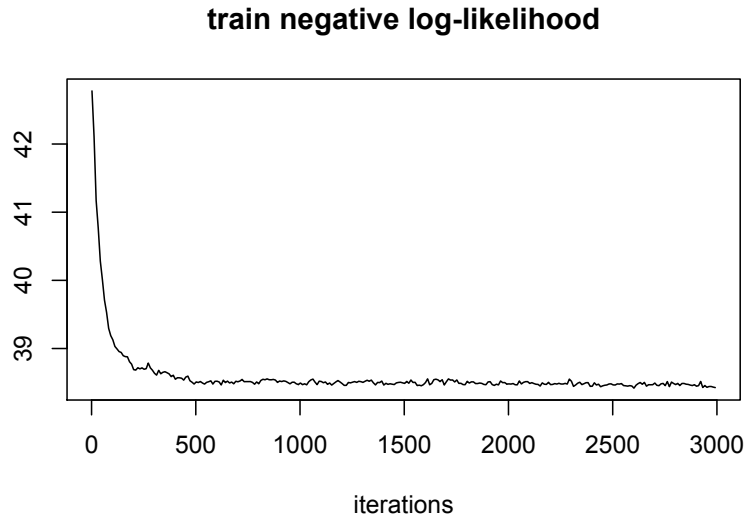


Figure 2.8: Convergence of the train negative log-likelihood.

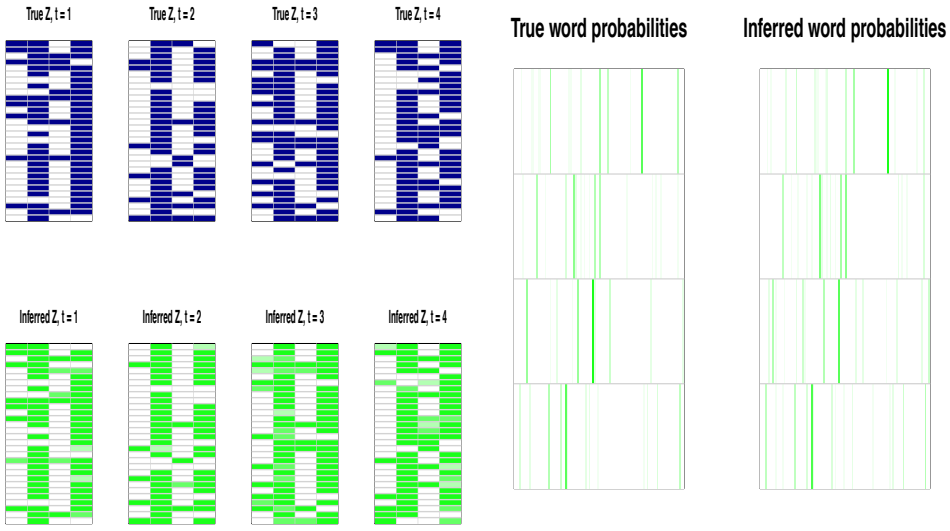


Figure 2.9: Left: Comparison between the true and the posterior mean topic allocation matrices at each time. Right: True vs inferred distributions over words for each topic. Each row is a topic ($K = 4$) and each column is a word from the dictionary ($D = 100$) (the darker the green, the larger the probability of the corresponding word).

this model observations at different time points are modeled separately, but information on the order of the time points is ignored.

In particular, we investigate whether incorporating time into the model improves upon test-set perplexity, a measure widely used in topic modeling settings

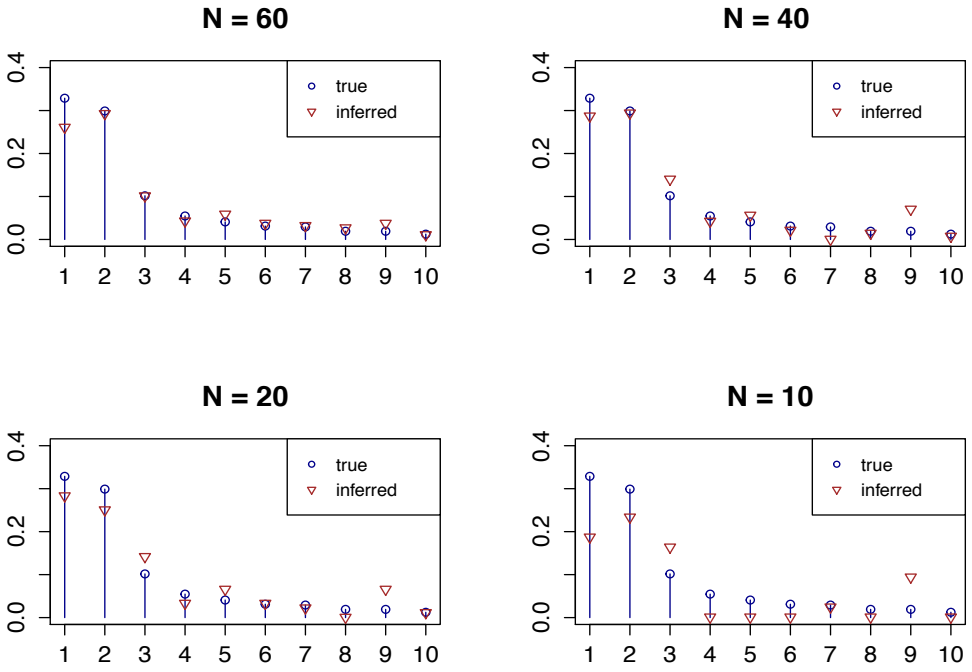


Figure 2.10: Comparison, given different numbers N of observed documents, between the true probabilities of the 10 most likely words within a given topic and the inferred probabilities of those words for that topic. The more documents are observed, the better the reconstruction of the topic is.

that assesses the ability of topic models to generalize to unseen data. Given the model parameters Φ, perplexity on documents $D_{test} := \{d_i\}_{i=1}^{M}$ is defined as

$$\text{perplexity}(D_{test} \mid \Phi) = \exp\left(-\frac{\sum_{i=1}^{M} \log p(d_i \mid \Phi)}{\sum_{i=1}^{M} W_i}\right),$$

where W_i denotes the number of words in document d_i. As we assume that words within each document are drawn independently given the model parameters Φ, the probability of document d_i can be computed as

$$p(d_i \mid \Phi) = \prod_{l=1}^{W_i} p(w_{il} \mid \Phi),$$

where

$$p(w_{il} \mid \Phi) = \sum_{k=1}^{K} \theta_{ik}\rho_{kl},$$

recalling that θ_{ik} is the probability of a generic word belonging to topic k in document i and ρ_{kl} is the probability of word l under topic k. These two quantities can be approximated at each iteration of the MCMC algorithm by their current values $\hat{\theta}_{ik}^{(s)}$ and $\hat{\rho}_{kl}^{(s)}$, so that we can approximate the probability of each word by averaging over S samples of the Markov Chain.

$$\hat{p}(w_{il} \mid \Phi) = \frac{1}{S} \sum_{s=1}^{S} \sum_{k=1}^{K} \hat{\theta}_{ik}^{(s)} \hat{\rho}_{kl}^{(s)}.$$

Note that the perplexity is inversely proportional to the likelihood of the data and thus lower values indicate better performance. Chance performance, namely assuming each word to be picked uniformly at random from the dictionary, yields a perplexity equal to the size D of the dictionary.

Different percentages of words were held-out and the model was trained on the remaining data. Testing the model on held-out words is a way to avoid comparing different hyper-parameters, as different treatments of the hyper-parameters could strongly affect the results [Asuncion et al., 2009]. For all three models, a dictionary of $D = 1000$ words was used to generate 30 documents at each of 9 time points. The number of features was fixed to 4 and the algorithms were run 3000 iterations with a burn-in period of 300 iterations. Even though all three algorithms approximately recover the true topic allocations matrices, incorporating time leads to a closer match. This can be measured, for instance, by the Frobenius norm of the difference between each true and inferred Z_t. Table 2.1 shows that at each time the dynamic model leads to a lower discrepancy with the true topic allocation matrix. Figure 2.11 shows the posterior trajectories of topic probabilities inferred by the dynamic model and compares them with the posterior feature probabilities inferred by the hierarchical model and the constant values inferred by the static model. It can be observed that the trajectories inferred by the dynamic model are both smoother and closer to the ground truth. Finally, Figure 2.12 compares the

test-set perplexity of the three models (recall that in this case chance performance results in a perplexity of $D = 1000$). As expected, although the hierarchical version performs better than the static counterpart that neglects time-stamp information, it is outperformed by our dynamic model. Indeed, while in the WF-IBP observations that are closer in time exhibit a stronger dependency, the HBP does not explicitly impose an ordering of the time points. The results show that having a suitable model for time dependencies improves on the ability to recover ground truth as well as to generalize to unseen data.

Table 2.1: Frobenius norm of the difference with the true Z_t. The lower, the better.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
Dynamic	**1.78**	**1.37**	**0.72**	**2.99**	**3.24**	**2.01**	**2.59**	**2.45**	**2.88**
Hierarchical	1.94	1.51	2.80	3.11	3.52	2.91	2.65	2.59	3.24
Static	3.63	4.23	2.84	3.01	3.55	5.27	4.89	3.31	5.90

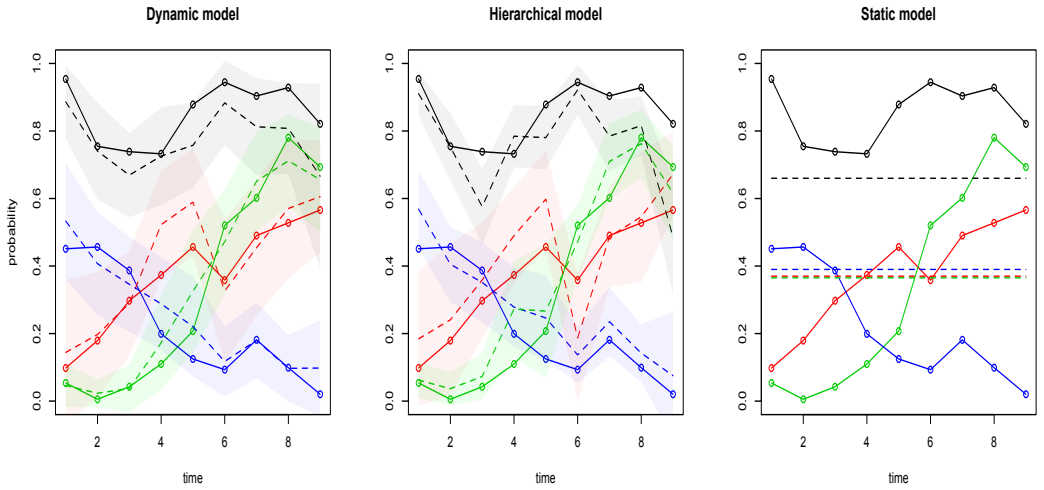


Figure 2.11: Comparison between true and inferred feature probabilities (respectively continuous and dotted lines) in our fixed-K topic model (left), in the hierarchical version (center) and in the static version (right).

Real-world data experiments

We used the WF-IBP topic model to explore the data set consisting of the full text of 5811 NIPS conference papers published between 1987 to 2015.[1] We pre-processed

[1]The data set is available at `https://archive.ics.uci.edu/ml/datasets/NIPS+Conference+Papers+1987-2015`.

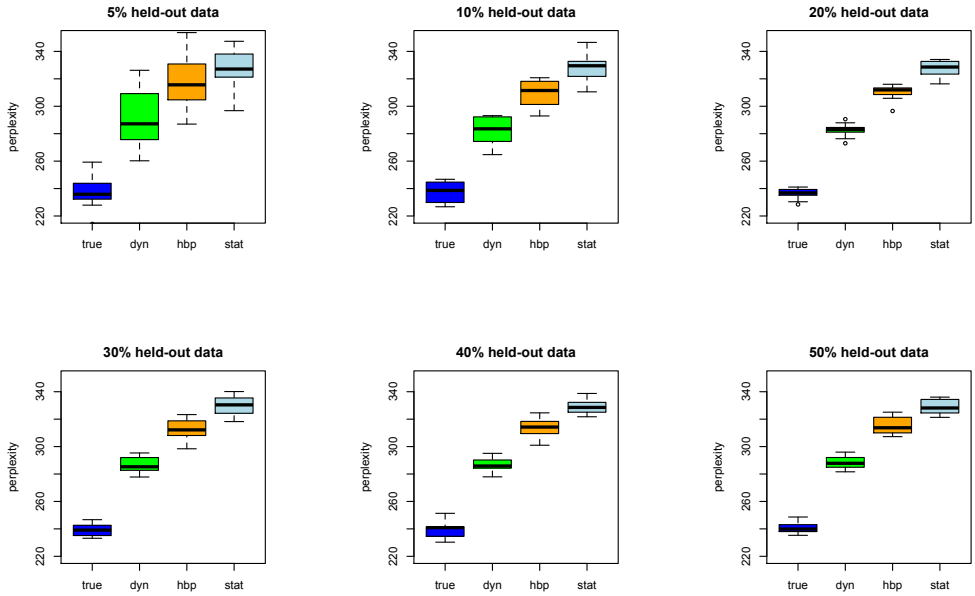


Figure 2.12: Boxplots of test-set perplexity for different percentages of held-out data for the true model (true), our dynamic model (dyn), the hierarchical version (hbp) and the static version (stat). Each boxplot was obtained by computing the perplexity after holding-out 10 different random subsets of words in the data. Lower values indicate better performance.

the data and removed words appearing more than 5000 times or fewer than 250 times. The remaining number of word tokens was 4 728 892 with a vocabulary size of 348 672 unique words. Our goal was to discover what topics appear in the corpus and to track the evolution of their popularity over these 29 years.

We set the hyperparameters α and β equal to 1 and the time step to 0.12 diffusion time-units per year so as to reflect realistic evolutions of topic popularity. The Markov chain was run for 2000 iterations with a burn-in period of 200 iterations, setting $\eta = 0.001$ and placing a Gamma(5,1) hyper-prior on γ.

Qualitative results One of the qualitative advantages of modeling time dependency explicitly is that interesting insights into the evolution of topics underlying large collections of documents can be obtained automatically, and uncertainty in the predictions naturally incorporated. The 12 most likely words of 32 topics found in the corpus together with the evolution of their topic proportions are given in Figure 2.13, where the shaded areas represent one standard deviation around the posterior means. As topics are defined by their distribution over words, it is possible to label them by looking at their most likely words. We observe that, with very few exceptions, the topics detected in the corpus are meaningful and easily interpretable. Figure 2.14 compares how the popularity of three different approaches to machine learning evolved over time. The results indicate that standard neural networks ('NNs backpropagation') were extremely popular until the early 90s. After this, they went through a steady decline, only to increase in popularity later on. This confirms the well known fact that NNs were largely forsaken in the machine learning community in the late 90s [see LeCun et al., 2015]. On the other hand, it can be observed that the popularity of deep architectures and convolutional neural networks ('deep learning') steadily increased over these 29 years, to the point that deep learning became the most popular among all topics in NIPS 2015.

Another key benefit of the WF-IBP over alternative nonparametric topic models is that the overall probability of topics and their proportion within documents are modeled separately, which allows rare topics to be the predominant subject within a few documents. This can be observed by comparing the rarest topic probabilities with the corresponding within-document topic proportions. In

a number of documents WF-IBP reveals that the predominant topic is among the rarest topics in the corresponding year, such as in "Text Classification using String Kernels" (information retrieval),"Playing is Believing: The Role of Beliefs in Multi-Agent Learning" (game theory), and "Relative Density Nets: A New Way to Combine Backpropagation with HMMs" (speech recognition).

We then compared the document representations learned by WF-IBP with the ones obtained by Dynamic Topic Models (DTM) [Blei and Lafferty, 2006]. One of the benefits of WF-IBP is that it provides sparse and less noisy representations. This is due to the fact that, while DTM assigns a positive probability to all topics within each document, the WF-IBP selects only a subset of topics with positive probability via the feature allocation matrices Z. Recall that DTM requires fixing the number of topics K a priori, hence in our experiments we set $K = 50$. For instance, "Recursive Training of 2D-3D Convolutional Networks for Neuronal Boundary Prediction", "Exploring Models and Data for Image Question Answering" and "Are You Talking to a Machine? Dataset and Methods for Multilingual Image Question Answering" are respectively assigned to deep learning, image recognition and NLP by both models; however, they are respectively explained by 9,8 and 10 topics in WF-IBP, as opposed to 50 in DTM. While it is possible to order the topic proportions in DTM and only consider the ones greater than an arbitrary threshold, WF-IBP automatically sets the probability of irrelevant topics to 0 and offers a more interpretable representation.

Quantitative results We then compared the predictive performance of our fixed-K approximation with its static and hierarchical counterparts. Recall that time-stamps are not used in the static model and their ordering is neglected in the hierarchical model. The results in Figure 2.15 were obtained by holding out different percentages of words (50%, 60%, 70% and 80%) from all the papers published in 1999 and by training the model over the papers published in the time range 1987-1999. The goal was to investigate whether incorporating time dependency improves the predictions on future documents at time $t+1$ when given the documents up to time t. The held-out words were then used to compute the test-set perplexity after 5 repeated runs with random initializations (the error bars represent one standard

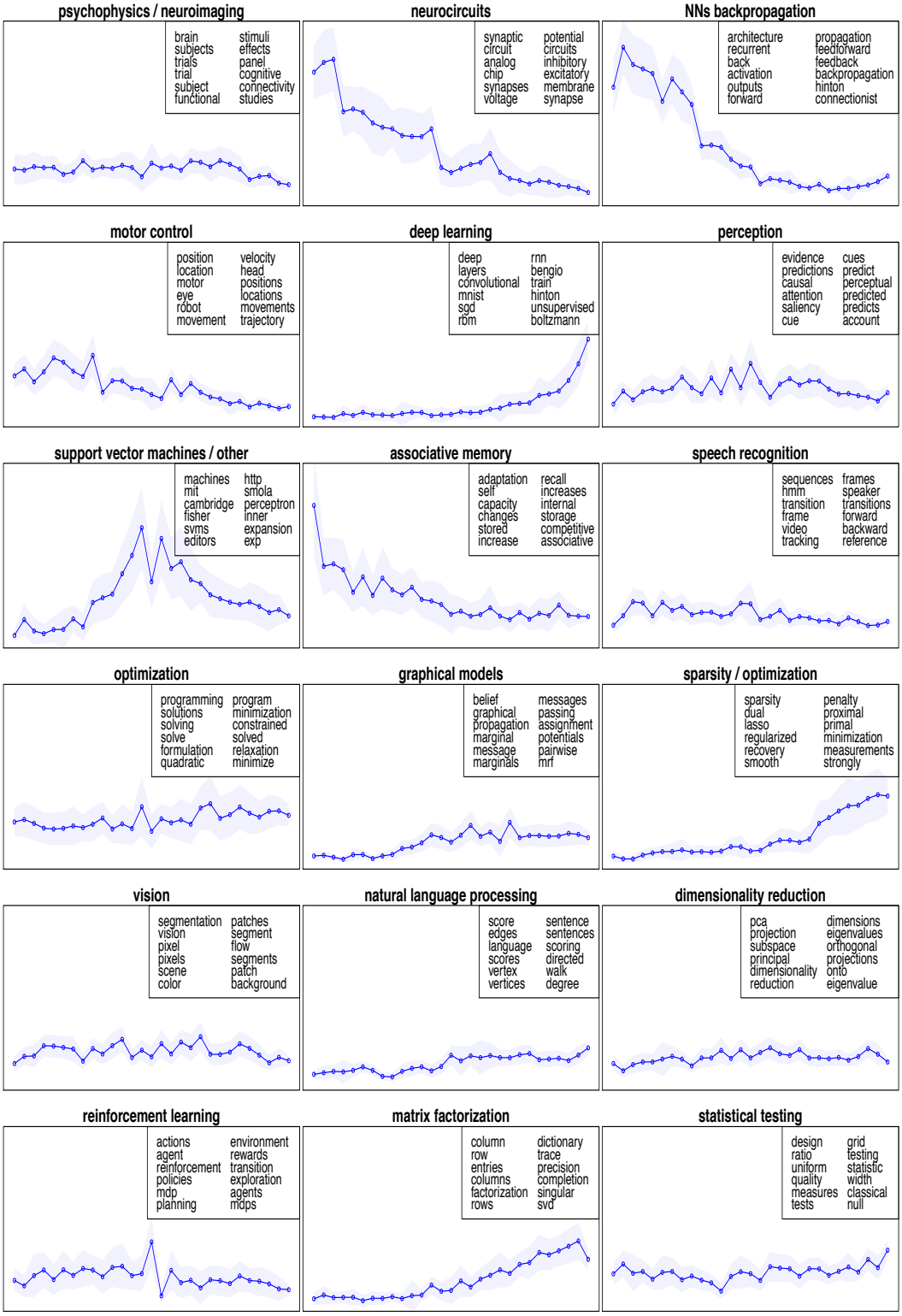
deviation). The dynamic model led to consistently better results, especially as the number of held-out words was increased. This follows from the fact that, the less training data is available in the year in which the models are tested, the more important capturing time-dependence to yield sensible predictions.

2.8 Discussion

We have presented a new framework for generating dependent IBPs by means of a novel time-evolving beta process, whereby feature probabilities evolve over time and are marginally distributed as in the beta process. At each time point items are exchangeable, and the two-parameter IBP is recovered. The key insight has been building on the PRF from population genetics to derive a suitable model for the prevalence and evolution of features over continuous time. We have developed an interesting MCMC framework for exact posterior inference with this model, and presented an alternative finite-dimensional approximation where the number of features is fixed.

As an application of the WF-IBP, we have described a time-dependent focused topic model that builds on Williamson et al. [2010b]. The WF-IBP topic model allows for a flexible evolution of the popularity of an unknown number of topics over time, and compares favorably to HDP models by decoupling topic probabilities and within-document topic proportions. We have used our model to explore the data set consisting of the full text of NIPS conference papers from 1987 to 2015 and obtained an interesting visualization of how the popularity of the underlying topics evolved over these 29 years. In addition, test-set perplexity results have shown that incorporating time also improves on the predictive performance of the model.

A number of directions for future work are open. First, as $K \to \infty$ the fixed-K approximation marginally converges to the infinite model, and simulations showed that their dynamics are remarkably similar; further work could formally investigate the exact relationship between the two dynamics. Second, the current MCMC framework could be generalized to include inference on the IBP parameters and the W-F diffusion time step. A straightforward extension could consider


psychophysics / neuroimaging
brain stimuli
subjects effects
trials panel
trial cognitive
subject connectivity
functional studies
neurocircuits
synaptic potential
circuit circuits
analog inhibitory
chip excitatory
synapses membrane
voltage synapse
NNs backpropagation
architecture propagation
recurrent feedforward
back feedback
activation backpropagation
outputs hinton
forward connectionist
motor control
position velocity
location head
motor positions
eye locations
robot movements
movement trajectory
deep learning
deep rnn
layers bengio
convolutional train
mnist hinton
sgd unsupervised
rbm boltzmann
perception
evidence cues
predictions predict
causal perceptual
attention predicted
saliency predicts
cue account
support vector machines / other
machines http
mit smola
cambridge perceptron
fisher inner
svms expansion
editors exp
associative memory
adaptation recall
self increases
capacity internal
changes storage
stored competitive
increase associative
speech recognition
sequences frames
hmm speaker
transition transitions
frame forward
video backward
tracking reference
optimization
programming program
solutions minimization
solving constrained
solve solved
formulation relaxation
quadratic minimize
graphical models
belief messages
graphical passing
propagation assignment
marginal potentials
message pairwise
marginals mrf
sparsity / optimization
sparsity penalty
dual proximal
lasso primal
regularized minimization
recovery measurements
smooth strongly
vision
segmentation patches
vision segment
pixel flow
pixels segments
scene patch
color background
natural language processing
score sentence
edges sentences
language scoring
scores directed
vertex walk
vertices degree
dimensionality reduction
pca dimensions
projection eigenvalues
subspace orthogonal
principal projections
dimensionality onto
reduction eigenvalue
reinforcement learning
actions environment
agent rewards
reinforcement transition
policies exploration
mdp agents
planning mdps
matrix factorization
column dictionary
row trace
entries precision
columns completion
factorization singular
rows svd
statistical testing
design grid
ratio testing
uniform statistic
quality width
measures classical
tests null

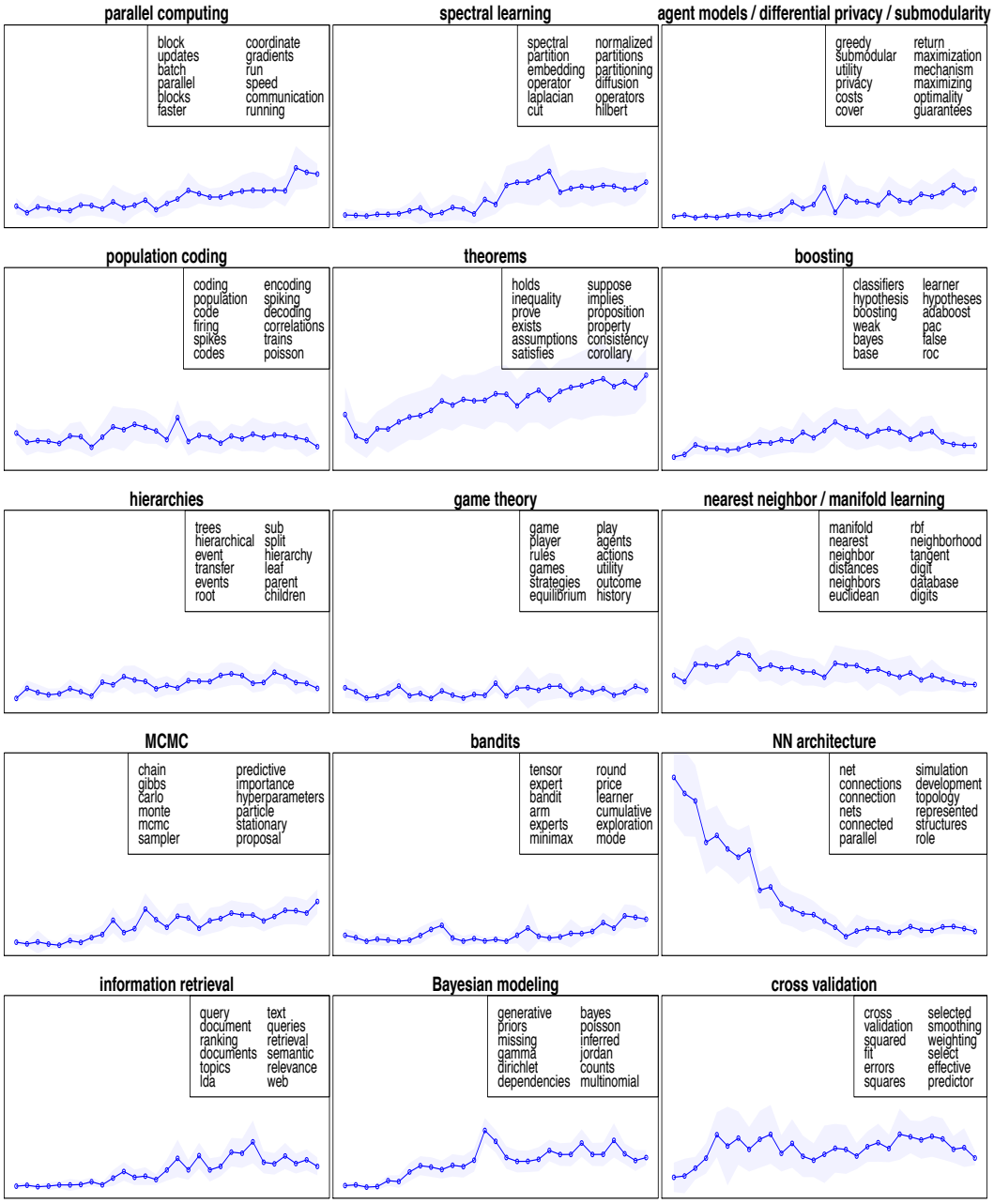


Figure 2.13: Posterior topic proportions over the years 1987-2015 and 12 most likely words for each topic (NIPS data set).

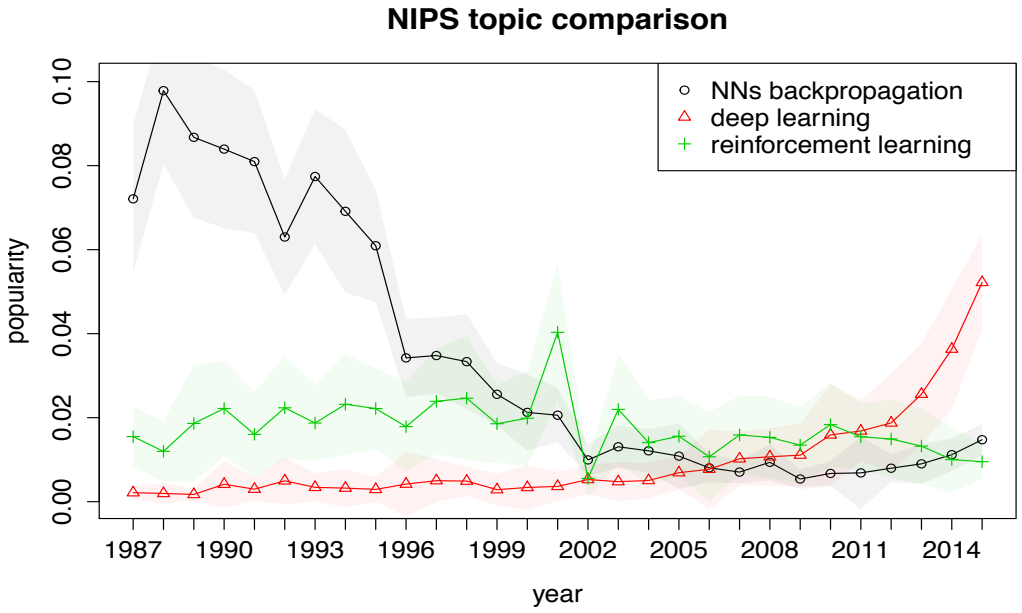


Figure 2.14: Comparison of a set of posterior topic proportions over the years 1987-2015 (NIPS data set).

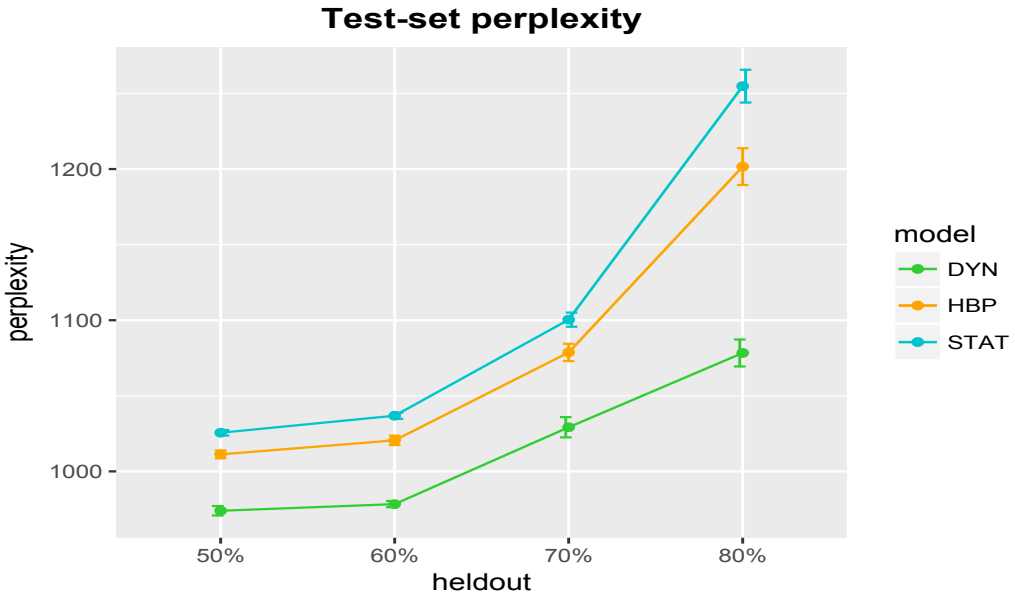


Figure 2.15: Comparison between the test-set perplexity of our dynamic model (DYN) against its hierarchical (HBP) and static (STAT) counterparts after holding out different percentages of words (NIPS data set). The dynamic model consistently outperforms the other two models, with a substantial difference when the percentage of held-out words is large.

an empirical Bayesian approach that learns the hyper-parameters by maximizing the likelihood of the observed data. Third, an extension of this work could mod-

ify the PRF by letting features evolve according to a more general W-F diffusion with selection and recombination, which would allow for feature-specific drifts in popularity and the coupled evolution of different features, respectively. Finally, our novel time-dependent beta process is a general construction with applications not limited to topic modeling. Different data and likelihood models could be explored following our work, for applications such as the modeling of time-evolving social networks or gene expression patterns.

2.9 Appendix

2.9.1 Appendix A. Proof of Theorem 1

The mean density of a PRF with immigration parameter α is

$$l(x) = \alpha m(x)dx,$$

where

$$m(x) = \frac{e^{I(x)}}{\sigma^2(x)}, \quad \text{with } I(x) := \int_0^x \frac{2\gamma(y)}{\sigma^2(y)}dy, \tag{2.11}$$

is the speed density of the process [see Griffiths, 2003]. γ and σ are the drift and diffusion terms as defined in (2.1) and (2.2). Plugging (2.1) with $\mu = 0$ and (2.2) into the integral, we have

$$I(x) = \beta \ln(1-x)$$

so that

$$m(x) = \frac{(1-x)^\beta}{x(1-x)} = x^{-1}(1-x)^{\beta-1}.$$

It follows that

$$l(x) = \alpha m(x)dx = \alpha x^{-1}(1-x)^{\beta-1}dx$$

is the resulting mean density, which completes the proof.

Remark: When $\mu, \beta = 0$ it is necessary to condition each diffusion on hitting the boundary 0 before 1 in reverse time, which leads to an extra term in the analogous result in Sawyer and Hartl [1992].

2.9.2 Appendix B. Simulating from the model

Consider the problem of simulating from the model, namely generating feature probabilities and the corresponding feature allocation matrices at a discrete set of

time points.

Simulating X Set a truncation level $u > 0$ and consider the task of simulating features whose probability is above the threshold u at two times t_0 and t_1. As we know how to simulate marginally from the beta process, we can first generate the feature probabilities above u at time t_0 and let them evolve independently to time t_1. This yields features whose probability is greater than u at time t_0, meaning that we are still missing those features whose probability is below u at time t_0. To simulated these, we proceed as follows: we generate these features by drawing them from the beta process at time t_1 and propagate them backwards to time t_0. Finally, in order not to double-count features, all features that in the reverse simulation have probability greater than u at time t_0 have to be rejected.

Now translate these ideas into the following sampling scheme. At time t_0, sample from a truncated version of the PRF, namely from a Poisson process on $[u, 1)$ with rate measure $\alpha x^{-1}(1-x)^{\beta-1}dx$. This can be done, for instance, via an adaptive thinning scheme as described in Ogata [1981]. As the truncation level u eliminates the point zero which has an infinite mass, this sampling procedure yields an almost surely finite number of samples. Denote by $\mathcal{K}$ the resulting set of feature indices and proceed as follows.

1. For all $k \in \mathcal{K}$, simulate $X_k(t) \mid \{X_k(t_0) = x_k(t_0)\} \sim \mathrm{WF}(0, \beta)$ for $t \in [t_0, t_1]$.

2. At time t_1, sample the candidate newborn features $X(t_1)$ from the truncated PRF as above. Let $\mathcal{L}$ denote the resulting set of features.

3. For all $l \in \mathcal{L}$, simulate $X_l(t) \mid \{X_l(t_1) = x_l(t_1)\} \sim \mathrm{WF}(0, \beta)$ backwards for $t \in [t_1, t_0]$ and remove from $\mathcal{L}$ the indices in the set $\{l : x_l(t_0) \geq u\}$.

4. Generate $Z_{ikt} \mid \{X_k(t) = x_k(t)\} \overset{iid}{\sim} \mathrm{Bernoulli}(x_k(t))$, for $t = t_0, t_1$, $\forall k \in \mathcal{K} \cup \mathcal{L}$ and $\forall i = 1, \ldots, N_t$.

As mentioned previously, the idea behind steps 2 and 3 is to compensate for the features with probability smaller than u that were discarded when generating features at time t_0.

This construction generalizes to a set of time points $t = t_0, \ldots, t_T$, observing that at a given time $t_{t^*} \in \{t_1, \ldots, t_T\}$ step 3 needs to be modified by simulating the W-F diffusions backwards to time t_0 and removing from $\mathcal{L}$ the indices such that $\exists t \in \{t_0, \ldots, t_{t^*-1}\}$ such that $x_l(t) \geq u$.

Simulating Z and the underlying X Consider now the more complex task of simulating both the feature allocation matrices Z and the features X appearing in them. First note that, although the PRF describes the evolution of an infinite number of features, we can sample the feature allocation matrices Z_{t_0} and Z_{t_1} at two times t_0 and t_1 exactly, as a property of the IBP is that the number of observed features is almost surely finite [Griffiths and Ghahramani, 2011]. It is then possible to sample the features that are active in at least one object at times t_0 and t_1 and the corresponding allocation matrices Z_{t_0} and Z_{t_1} as follows. First, draw Z_{t_0} from the IBP, and use its realisation to draw the posterior probabilities $X(t_0)$ of the features seen in Z_{t_0} as in the posterior beta process. Then, simulate from the W-F diffusion to propagate these features to time t_1 and generate Z_{t_1} using these feature probabilities. We are now only missing the columns of Z_{t_1} corresponding to the features that were seen at time t_1 but not at time t_0. To add those columns, first draw a candidate $Z^C_{t_1}$ from the IBP and the corresponding feature probabilities; then, simulate these candidate features backwards to time t_0 and accept them with probability $(1-X(t_0))^{N_{t_0}}$ to account for the fact that they were not seen at time t_0. The columns of $Z^C_{t_1}$ corresponding to the rejected features are deleted. Translating these ideas into an algorithm, consider the following steps.

1. Draw $Z_{t_0} \sim \text{IBP}(\alpha, \beta)$ and index the resulting columns as $1, \ldots, K_1$.

2. For $k = 1, \ldots, K_1$ draw the corresponding feature probabilities

$$X_k(t_0) \mid \{Z_{t_0} = z_{t_0}\} \sim \text{Beta}\left(n_{kt_0}, \beta + N_{t_0} - n_{kt_0}\right).$$

3. For $k = 1, \ldots, K_1$ simulate $X_k(t) \mid \{X_k(t_0) = x_k(t_0)\} \sim \text{WF}(0, \beta)$ for $t \in [t_0, t_1]$ and set $X_k(t_1) = x_k(t_1)$.

4. Sample $Z^1_{t_1}$ by drawing each $Z^1_{ikt_1} \mid \{X(t_1) = x(t_1)\} \overset{iid}{\sim} \text{Bernoulli}(x_k(t_1))$,

where $k = 1, \ldots, K_1$ and $i = 1, \ldots, N_{t_1}$.

Then, to sample the features that are active only at time t_1, add the following steps.

5. Draw a candidate $Z^{\mathrm{C}}_{t_1} \sim \mathrm{IBP}(\alpha, \beta)$ and index the resulting columns as $K_1 + 1, \ldots, K_2$.

6. For $k = K_1+1, \ldots, K_2$ draw the corresponding candidate feature probabilities

$$X^{\mathrm{C}}_k(t_1) \mid \{Z^{\mathrm{C}}_{t_1} = z^{\mathrm{C}}_{t_1}\} \sim \mathrm{Beta}\left(n_{kt_1}, \beta + N_{t_1} - n_{kt_1}\right).$$

7. For $k = K_1 + 1, \ldots, K_2$, simulate $X_k(t) \mid \{X_k(t_1) = x_k(t_1)\} \sim \mathrm{WF}(0, \beta)$ backwards for $t \in [t_1, t_0]$ and set $X_k(t_0) = x_k(t_0)$.

8. Accept the candidate columns of $Z^C_{t_1}$ with probability $(1 - x^C_k(t_0))^{N_{t_0}}$ and let Z_{t_1} be the matrix obtained by the union of the columns of $Z^1_{t_1}$ with the accepted columns of $Z^C_{t_1}$.

Note that the rejection in the last step is a way to account for the fact that we are considering features that are active for the first time at time t_1. When considering a general set of time points $t_0, \ldots, t_T$, it is necessary to account for the features that are active for the first time at each time $t_1, \ldots, t_T$. In this more general case, features seen for the first time at time t_{t^*} need to be accepted with probability $\prod_{t=t_0}^{t_{t^*-1}} (1 - x^C_k(t))^{N_t}$, as they were not seen in any of the feature allocation matrices at the time points before t^*.

2.9.3 Appendix C. Derivation of full conditionals

As observed in Williamson et al. [2009], in this topic modeling setting there are two equivalent ways of generating documents. Either the total number of words is sampled from a negative binomial $\mathrm{NB}(\sum_k z_{ikt}\phi_{kt}, 1/2)$ and then the topic and word assignments are drawn, or the number of words generated by each topic is drawn from $\mathrm{NB}(z_{ikt}\phi_{kt}, 1/2)$ and then the word assignments are picked. Following Williamson et al. [2009] closely, we make use of the latter construction to derive the full conditional distributions for the Gibbs sampler of the WF-IBP topic model.

Full conditional of a_{il} Recall that $a_{ilt} = k$ indicates that the lth word in document i at time t is assigned to topic k. We have

$$\begin{aligned} p(a_{ilt} = k \mid a_{-il}, Z_t, w_{ilt}, \phi_t) &\propto p(w_{ilt} \mid a_{ilt} = k) p(a_{ilt} = k \mid a_{-il}, z_{ikt}, \phi_{kt}) \\ &\propto p(w_{ilt} \mid a_{ilt} = k)(n^i_{kt} + \phi_{kt} z_{ikt}), \end{aligned}$$

where the last step is given by integrating out θ_{it}, namely the distribution over topics in document i at time t, and using the Dirichlet-Categorical conjugacy. Recall that n^i_{kt} denotes the number of words assigned to topic k in document i at time t and that n_{kt} denotes the total number of words assigned to topic k at time t, both excluding the assignment a_{il}. Similarly, integrating out the parameter ρ_k representing the distribution over words of topic k and using the Dirichlet-Categorical conjugacy, we have that

$$p(w_{ilt} \mid a_{ilt} = k) = \frac{(n^{w_{il}}_k + \eta)}{n_k + \eta D - 1},$$

which, plugged into the previous equation, gives the desired full conditional.

Full conditional of ϕ_k and γ We have that

$$\begin{aligned} p(\phi_{kt}, \gamma \mid n_{kt}, X(t), Z_t) &\propto p(\phi_{kt}, \gamma, n_{kt}, X(t), Z_t) \\ &\propto p(\phi_{kt} \mid \gamma) P(\gamma) P(n_{kt} \mid Z_t, \phi_{kt}), \end{aligned} \tag{2.12}$$

where

$$p(n_{kt} \mid Z_t, \phi_{kt}) = \prod_{i=1}^{N_t} p(n^i_{kt} \mid Z_{ikt}, \phi_{kt}) = \prod_{i=1}^{N_t} \mathrm{NB}(n^i_{kt}; Z_{ikt}\phi_{kt}, 1/2).$$

Note that $p(\phi_{kt} \mid \gamma)$ is distributed according to its prior $\mathrm{Gamma}(\gamma, 1)$ and γ according to a chosen hyper-prior. The result follows immediately by plugging these three distributions into (2.12).

Full conditional of Z_{ikt} Recall that n_{ikt} denotes the total number of words assigned to topic k in document i at time t. If $n_{ikt} > 0$, then the corresponding

entry Z_{ikt} is active with probability 1. If $n_{ikt} = 0$, we have

$$p(Z_{ikt} = 1 \mid Z_{-(ik)t}, n_{ikt} = 0, X_k(t), \phi_{kt}, S_t) = \frac{p(Z_{ikt} = 1, Z_{-(ik)t}, n_{ikt} = 0, X_k(t), \phi_{kt}, S_t)}{p(Z_{-(ik)t}, n_{ikt} = 0, X_k(t), \phi_{kt}, S_t)}.$$

The numerator is equal to

$$p(n_{ikt} = 0 \mid Z_{ikt} = 1, \phi_{kt}) p(S_t \mid Z_{ikt} = 1, Z_{-(ik)t})$$
$$p(Z_{ikt} = 1 \mid X_k(t)) p(\phi_{kt}, X_k(t), Z_{-(ik)t}) =$$
$$\mathrm{NB}(0; \phi_{kt}, 1/2) \frac{1}{x_1^{min}(t)} x_k(t) p(\phi_{kt}, X_k(t), Z_{-(ik)t})$$

Denoting by C the product of all terms not depending on z_{ikt}, we have

$$p(Z_{ikt} = 1 \mid Z_{-(ik)t}, n_{ikt} = 0, X_k(t), \phi_{kt}, S_t) = C \frac{1}{2^{\phi_{kt}}} \frac{1}{x_1^{min}(t)} x_k(t). \tag{2.13}$$

By the same token, we have

$$p(Z_{ikt} = 0 \mid Z_{-(ik)t}, n_{ikt} = 0, X_k(t), \phi_{kt}, S_t) = C \frac{1}{x_0^{min}(t)} (1 - x_k(t)). \tag{2.14}$$

As the two probabilities must sum to 1, we have that

$$C = \frac{2^{\phi_{kt}} x_1^{min}(t) x_0^{min}(t)}{x_0^{min}(t) x_k(t) + 2^{\phi_{kt}} x_1^{min}(t)(1 - x_k(t))},$$

which, plugged into equations 2.13 and 2.14, gives the result.

3

Scalable and Robust Bayesian Sampling

In the previous chapter, we developed a sophisticated MCMC algorithm that combines slice sampling with Particle Gibbs to achieve exact posterior inference. Another class of MCMC techniques to efficiently sample from posterior distributions is based on the simulation of continuous-time physical systems. Hamiltonian Monte Carlo (HMC) is a widely-used MCMC algorithm that generates proposals for a Metropolis-Hastings algorithm by simulating the dynamics of a Hamiltonian system. In the complex and large-scale settings we consider in this thesis, however, HMC can perform poorly if there is a mismatch between the spatial geometry of the target distribution and the scales of the momentum distribution. In particular, results can very sensitive to the choice of the mass matrix of HMC.

In this chapter, we aim to alleviate these issues by proposing relativistic Hamiltonian Monte Carlo, a version of HMC based on the relativistic dynamics that introduces a maximum velocity on particles. We derive stochastic gradient versions of the algorithm and show that the resulting algorithms bear interesting relationships with gradient clipping, RMSprop, Adagrad and Adam, popular optimization methods in deep learning. In experiments, we show that the relativistic algorithms are much more robust to the choice of parameter settings, outperforming classical Newtonian variants.

3.1 Introduction

MCMC methods based on the simulation of dynamical systems are a mainstay of Bayesian machine learning and statistics. Hamiltonian Monte Carlo (HMC) [Duane et al., 1987; Neal, 2010; Carpenter, 2015; Hoffman and Gelman, In press] is based on the Newtonian dynamics on a frictionless surface, and has been argued to be more efficient than techniques based on diffusions [Xifara et al., 2014]. On the other hand, stochastic gradient MCMC techniques based on diffusive dynamics [Welling and Teh, 2011; Ma et al., 2015; Ding et al., 2014; Chen et al., 2014] made it possible for models to be trained on mini-batches of the data, scaling up Bayesian inference.

An important consideration when designing such MCMC algorithms is adaptation or tuning to the geometry of the space under consideration [Girolami and Calderhead, 2011; Beskos et al., 2013; Patterson and Teh, 2013]. To give a concrete example, consider HMC. Let $f(\theta)$ be a target density which can be written as $f(\theta) \propto e^{-U(\theta)}$ where $U(\theta)$ is interpreted as the potential energy of a particle in location θ. HMC introduces an auxiliary momentum variable p so that the joint distribution is $f(\theta, p) \propto e^{-H(\theta,p)}$ where the Hamiltonian is $H(\theta, p) = U(\theta) + \frac{1}{2m} p^\top p$. The quantity $\frac{1}{2m} p^\top p$, where m is the mass of the particle, represents the kinetic energy. Denoting by $\dot{\theta}$ and $\dot{p}$ the time derivative of θ and p, the leapfrog discretisation [Neal, 2010] of Hamilton's equations $\dot{\theta} = \frac{\partial H}{\partial p}$ and $\dot{p} = -\frac{\partial H}{\partial \theta}$ gives

$$
\begin{aligned}
p_{t+1/2} &\leftarrow p_t - \tfrac{1}{2}\epsilon \nabla U(\theta_t), \\
\theta_{t+1} &\leftarrow \theta_t + \epsilon \tfrac{p_{t+1/2}}{m}, \\
p_{t+1} &\leftarrow p_{t+1/2} - \tfrac{1}{2}\epsilon \nabla U(\theta_{t+1}),
\end{aligned}
$$

where ϵ is the time discretisation and the velocity is $\frac{p_{t+1/2}}{m}$. If m is too small, the particle travels too fast leading to an accumulation of discretisation error. To compensate, ϵ needs to be set small and the computational cost required increases. On the other hand, if m is too large, the particle travels slowly resulting in slow mixing of the resulting Markov chain. While the mass parameter can be tuned, e.g. to optimise acceptance rate according to theory [Beskos et al., 2013], it only incidentally controls the velocity which ultimately affects the discretisation error

and algorithm stability.

In this chapter, we are interested in making MCMC algorithms based on physical simulations more robust by directly controlling the velocity of the particle. This is achieved by replacing Newtonian dynamics in HMC with relativistic dynamics [Einstein, 1905], whereby particles cannot travel faster than the "speed of light". We also develop relativistic variants of stochastic gradient MCMC algorithms and show that they work better and are more robust than the classical Newtonian variants.

The relativistic MCMC algorithms we develop have interesting relationships with a number of optimisation algorithms popular in deep learning. Firstly, the maximum allowable velocity (speed of light) is reminiscent of gradient clipping [Pascanu et al., 2013]. Our framework gives Bayesian alternatives to gradient clipping, in the sense that our algorithms demonstrably sample from instead of optimising the target distribution (exactly or approximately). Secondly, the resulting formulas (see (3.3)), which include normalisations by L_2 norms, bear strong resemblances to (but are distinct from) RMSprop, Adagrad and Adam [Tieleman and Hinton, 2012; Duchi et al., 2011; Kingma and Ba, 2014c]. Motivated by these connections, Xiaoyu Lu (co-author) develops a relativistic stochastic gradient descent (SGD) algorithm by taking the zero-temperature limit of relativistic SGHMC, showing that it outperforms Adam in experiments with neural networks.

3.2 Relativistic Hamiltonian Dynamics

Our starting point is the Hamiltonian which governs dynamics in special relativity [Einstein, 1905],

$$H(\theta, p) = U(\theta) + K(p), \tag{3.1}$$

$$K(p) = mc^2 \left(\frac{p^\top p}{m^2 c^2} + 1 \right)^{\frac{1}{2}} \tag{3.2}$$

where the target density is

$$f(\theta) \propto e^{-U(\theta)}$$

for $\theta \in \mathbb{R}^d$ interpreted as the position of the particle, $p \in \mathbb{R}^d$ is a momentum variable, and $K(p)$ is the relativistic kinetic energy. The two tunable hyperparameters are a scalar 'rest mass' m and the 'speed of light' c which bounds the particle's speed. The joint distribution $f(\theta, p) \propto e^{H(\theta,p)}$ is separable, with the momentum variable having marginal distribution $\propto e^{-K(p)}$, a multivariate generalisation of the symmetric hyperbolic distribution.

The resulting dynamics are given by Hamilton's equations, which read

$$\begin{aligned} \dot{\theta} &= \frac{\partial H}{\partial p} = M^{-1}(p)p, & M(p) &= m\left(\frac{p^\top p}{m^2c^2} + 1\right)^{\frac{1}{2}} \\ \dot{p} &= -\frac{\partial H}{\partial \theta} = -\nabla U(\theta), \end{aligned} \tag{3.3}$$

where $M(p)$ can be interpreted as a relativistic mass and $M^{-1}(p)p$ is the velocity of the particle (c.f. the velocity under Newtonian dynamics is $m^{-1}p$). Note that the relativistic mass is lower bounded by and increases asymptotically to $\|p\|/c$ as the momentum increases, so that the speed $M^{-1}(p)\|p\|$ is upper bounded by and asymptototes to c. On the other hand, the larger the rest mass m the smaller the typical "cruising" speed of the particle is. Conversely, as $m \to 0$ the particle will travel at the speed of light at all times, i.e. it behaves like a photon. This gives an intuition for tuning both hyperparameters c and m based on knowledge about the length scale of the target density: we choose c as an upper bound on the speed at which the parameter of interest θ changes at each iteration, while we choose m to control the typical sensible speed at which the parameter changes. We will demonstrate this intuition in the experimental Section 3.4.

In very high dimensional problems (e.g. those in deep learning, collaborative filtering or probabilistic modelling), the maximum overall speed imposed on the system might need to be very large so that reasonably large changes in each coordinate are possible at each step of the algorithm. This means that each coordinate could in principle achieve a much higher speed than desirable. An alternative approach is to upper bound the speed at which each coordinate changes by choosing

the following relativistic kinetic energy

$$K(p) = \sum_{j=1}^{d} m_j c_j^2 \left(\frac{p_j^2}{m_j^2 c_j^2} + 1 \right)^{\frac{1}{2}}, \tag{3.4}$$

where j indexes the coordinates of the d-dimensional system, and each coordinate can have its own mass m_j and speed of light c_j. This leads to the same Hamiltonian dynamics (3.3), but with all variables interpreted as vectors, and all arithmetic operations interpreted as element-wise operations. Experimental results will be based on the separable variant which showed consistently better performance. For simplicity, in the theoretical sections we will describe only the non-separable version (3.2).

3.2.1 Relativistic Hamiltonian Monte Carlo

As a demonstration of the relativistic Monte Carlo framework, we derive a relativistic variant of the Hamiltonian Monte Carlo (HMC) algorithm [Neal, 2010; Duane et al., 1987]. In the following, we will refer to all classical variants as Newtonian as they follow Newtonian dynamics, namely *Newtonian HMC (NHMC)* vs *relativistic HMC (RHMC)*.

Each iteration of HMC involves first sampling the momentum variable, followed by a series of L leapfrog steps, followed by a Metropolis-Hastings accept/reject step. The momentum can be simulated by first simulating the speed $\|p\|$ followed by simulating p uniformly distribution on the sphere with radius $\|p\|$. The speed $\|p\|$ has marginal distribution given by a symmetric hyperbolic distribution, for which specialised random variate generators exist. Alternatively, the density is log-concave, and we used adaptive rejection sampling to simulate it. The leapfrog steps [Calvo, 1994] with stepsize ϵ follows (3.3) directly: Set θ_0, p_0 to the current location and momentum and for $t = 1, \ldots, L$,

$$\begin{aligned} p_{t+1/2} &\leftarrow p_t - \tfrac{1}{2}\epsilon \nabla U(\theta_t) \\ \theta_{t+1} &\leftarrow \theta_t + \epsilon M^{-1}(p_{t+1/2}) p_{t+1/2} \\ p_{t+1} &\leftarrow p_{t+1/2} - \tfrac{1}{2}\epsilon \nabla U(\theta_{t+1}). \end{aligned} \tag{3.5}$$

The leapfrog steps leave the Hamiltonian H approximately invariant and is volume-preserving [Leimkuhler and Shang, 2016], so that the MH acceptance probability is simply $\min(1, \exp(-H(\theta_L, p_L) + H(\theta_0, p_0)))$.

Observe that the momentum p is unbounded and may become very large in the presence of large gradients in the potential energy. However, the size of the θ update is bounded by ϵc and therefore the stability of the proposed sampler can be controlled. This behaviour is essential for good algorithmic performance on complex models such as neural networks, where the scales of gradients can vary significantly across different parameters and may not be indicative of the optimal scales of parameter changes. This is consistent with past experiences optimising neural networks, where it is important to adapt the learning rates individually for each parameter so that typical parameter changes stay in a sensible range [Pascanu et al., 2013; Tieleman and Hinton, 2012; Duchi et al., 2011; Kingma and Ba, 2014c; Şimşekli et al., 2016]. Such adaptation techniques have also been explored for stochastic gradient MCMC techniques [Li et al., 2015; Teh et al., 2015], but we will argue that they introduce another form of instability that is not present in the relativistic approach.

3.3 Relativistic Stochastic Gradient Markov Chain Monte Carlo

In recent years stochastic gradient MCMC (SGMCMC) algorithms have been very well explored as methods to scale up Bayesian learning by using mini-batches of data [Welling and Teh, 2011; Chen et al., 2014; Ding et al., 2014; Ma et al., 2015; Shang et al., 2015]. In this section we develop relativistic variants of SGHMC [Chen et al., 2014] and SGNHT [Ding et al., 2014; Shang et al., 2015]. These algorithms include momenta, which serve as reservoirs of previous gradient computations, thus can integrate and smooth out gradient signals from previous mini-batches of data. As noted earlier, because the momentum can be large, particularly as the stochastic gradients can have large variance, the resulting updates to θ can be overly large, and small values of the step size are required for stability, leading potentially to slower convergence. This motivates our development of relativistic variants.

We make use of the framework of [Ma et al., 2015] for deriving SGMCMC algorithms. Let z be a collection of variables with target distribution $f(z) \propto e^{-H(z)}$. Consider an SDE in the form

$$dz = -[D(z) + Q(z)]\nabla H(z)dt + \Gamma(z)dt + \sqrt{2D(z)}dW, \tag{3.6}$$
$$\Gamma_i(z) = \sum_{j=1}^{d} \frac{\partial[D_{ij}(z)+Q_{ij}(z)]}{\partial z_j}$$

where $D(z)$ is a symmetric positive-definite diffusion matrix, $Q(z)$ is a skew-symmetric matrix which describes energy-conserving dynamics, $\Gamma(z)$ is a correction factor, and W is the d-dimensional Wiener process (Brownian motion). Ma et al. [2015] showed that under mild conditions the SDE converges to the desired stationary distribution $f(z)$. Hence in the following we simply have to choose the appropriate z, D and Q. Once the correction factor Γ is computed, the SDE discretised, and a stochastic estimate $\nabla\tilde{U}(z)$ for $\nabla U(z)$ substituted, we obtain a correct relativistic SGMCMC algorithm. The stochastic gradient has asymptotically negligible variance compared to the noise injected by W.

3.3.1 Relativistic Stochastic Gradient Hamiltonian Monte Carlo

Suppose our noisy gradient estimate $\nabla\tilde{U}(\theta)$ of $\nabla U(\theta)$ is based on a minibatch of data. Then, appealing to the central limit theorem, we can assume that $\nabla\tilde{U}(\theta) \approx \nabla U(\theta) + \mathcal{N}(0, B(\theta))$. Let $z = (\theta, p)$ and $H(z)$ be the relativistic Hamiltonian in (3.2). Choosing

$$D(z) = \begin{pmatrix} 0 & 0 \\ 0 & D \end{pmatrix}, Q(z) = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix}, \text{and thus } \Gamma(z) = \mathbf{0}, \tag{3.7}$$

where D is a fixed symmetric diffusion matrix results in the following relativistic SGHMC dynamics:

$$\begin{pmatrix} d\theta \\ dp \end{pmatrix} = \begin{pmatrix} M^{-1}(p)p \\ -\nabla U(\theta) - DM^{-1}(p)p \end{pmatrix} dt + \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{2D} \end{pmatrix} dW_t.$$

Using a simple Euler-Maruyama discretisation, the relativistic SGHMC algorithm is,

$$
\begin{aligned}
p_{t+1} &\leftarrow p_t - \epsilon_t \nabla \tilde{U}(\theta_t) - \epsilon_t D M^{-1}(p_t) p_t + \mathcal{N}(0, \epsilon_t(2D - \epsilon_t \hat{B}_t)) \\
\theta_{t+1} &\leftarrow \theta_t + \epsilon_t M^{-1}(p_{t+1}) p_{t+1}
\end{aligned} \tag{3.8}
$$

where $\hat{B}$ is an estimate of the noise coming from the stochastic gradient $B(\theta)$. The term $DM^{-1}(p)p$ can be interpreted as friction, which prevents the kinetic energy to build up and corrects for the noise coming from the stochastic gradient.

It is useful to compare RSGHMC with preconditioned SGLD [Li et al., 2015; Teh et al., 2015] which attempt to adapt the SGLD algorithm to the geometry of the space, using adaptations similar to RMSProp, Adagrad or Adam. The relevant term above is the update $M^{-1}(p_{t+1})p_{t+1}$ to θ_{t+1}:

$$
M^{-1}(p_{t+1}) p_{t+1} = \frac{p_{t+1}}{\sqrt{\frac{p_{t+1}^\top p_{t+1}}{c^2} + m^2}}. \tag{3.9}
$$

Note the surprising similarity to RMSProp, Adagrad and Adam, with the main difference being that the relativistic mass adaptation uses the current momentum instead of being separately estimated using the square of the gradient. This has the advantage that the relativistic SGHMC enforces a maximum speed of change. In contrast, preconditioned SGLD has the following failure mode which we observe in Section 3.4: when the gradient is small, the adaptation scales up the gradient so that the gradient update has a reasonable size. However it also scales up the injected noise, which can end up being significantly larger than the gradient update, and making the algorithm unstable.

3.3.2 Relativistic Stochastic Gradient Descent (with Momentum)

Motivated by the relationship to RMSprop, Adagrad and Adam, we develop a relativistic stochastic gradient descent (RSGD) algorithm with momentum by taking the zero-temperature limit of the RSGHMC dynamics. This formulation was derived by Xiaoyu Lu, co-author. This idea connects to Santa [Chen et al., 2016], a recently developed algorithm where an annealing scheme on the system tempera-

ture makes it possible to obtain a stochastic optimization algorithm starting from a Bayesian one.

From thermodynamics [Laurendeau, 2005], the canonical (Gibbs Boltzmann) density is proportional to $e^{-\beta U(z)}$ where $\beta = 1/k_B T$, k_B begin the Boltzmann constant and T the temperature. Previously we have been using $\beta = 1$ which corresponds to the posterior distribution. For general β,

$$\begin{pmatrix} d\theta \\ dp \end{pmatrix} = \begin{pmatrix} \beta M^{-1}(p)p \\ \beta\left(-\nabla U(\theta) - DM^{-1}(p)p\right) \end{pmatrix} dt + \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{2D} \end{pmatrix} dW \tag{3.10}$$

By taking $\beta \to \infty$ the target distribution becomes more peaked around the MAP estimator. Simulated annealing [Geman and Hwang, 1986; Andrieu and Doucet, 2000; Chen et al., 2016], which increases $\beta \to \infty$ over time, forces the sampler to converge to a MAP estimator. Instead, we can derive RSGD by rescaling time as well, guaranteeing a non-degenerate limit process. Letting $\hat{\theta}(t) = \theta(\beta t)$, $\hat{p}(t) = p(\beta t)$, so that

$$\begin{pmatrix} d\hat{\theta} \\ d\hat{p} \end{pmatrix} = \begin{pmatrix} M^{-1}(\hat{p})\hat{p} \\ -\nabla U(\hat{\theta}) - DM^{-1}(\hat{p})\hat{p} \end{pmatrix} dt + \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{\frac{2D}{\beta}} \end{pmatrix} dW \tag{3.11}$$

and letting $\beta \to \infty$, we obtain the following ODE:

$$\begin{pmatrix} d\theta \\ dp \end{pmatrix} = \begin{pmatrix} M^{-1}(p)p \\ -\nabla U(\theta) - DM^{-1}(p)p \end{pmatrix} dt. \tag{3.12}$$

Discretising the above then gives RSGD. Notice that if the above converges, i.e. $\dot{\theta} = \dot{p} = 0$, it does so at a critical point of U. Similar to other adaptation schemes, RSGD adaptively rescales the learning rates for different parameters, which enables effective learning especially in high dimensional settings. Moreover, the update in each iteration is upper bounded by the speed of light. Our algorithm differs from others through the use of a momentum, and adapting based on the momentum instead of the average of squared gradients.

3.3.3 A Stochastic Gradient Nosé-Hoover Thermostat for Relativistic Hamiltonian Monte Carlo

Borrowing a second concept from physics, SGHMC can be improved by introducing a dynamic variable ξ that adaptively increases or decreases the momenta. The new variable ξ can be thought of as a *thermostat* in a statistical physics setting and its dynamics expressed as

$$d\xi = \frac{1}{d}\left(p^T p - d\right) dt. \tag{3.13}$$

The idea is that the system adaptively changes the friction for the momentum, 'heating' or 'cooling down' the system. The dynamics of this new variable, known as Nosé-Hoover [Leimkuhler and Reich, 2009] thermostat due to its links to statistical physics, has been shown to be able to remove the additional bias due to the stochastic gradient provided that the noise is isotropic Gaussian and spatially constant ([Ding et al., 2014],[Leimkuhler and Shang, 2016]). In general, the noise is neither Gaussian, spatially constant or isotropic. Nevertheless, there is numerical evidence that the thermostat increases stability and mixing. Heuristically, the dynamic for ξ can be motivated as follows. At equilibrium we have

$$\mathbb{E}\left[\frac{\partial^2 K}{\partial p_i^2}\right] = \int \frac{\partial^2 K}{\partial p_i^2} e^{-K(p)} dp = -\int \frac{\partial K}{\partial p_i}\left(-\frac{\partial K}{\partial p_i} e^{-K(p)}\right) dp = \mathbb{E}\left[\left(\frac{\partial K}{\partial p_i}\right)^2\right]$$

and hence $\mathbb{E}\left[\frac{d\xi}{dt}\right] = 0$ (as derived by Leonard Hasenclever, co-author). The additional dynamics pushes the system towards $\frac{d\xi}{dt} = 0$ suggesting that the distribution will be moved closer to the equilibrium. This gives a recipe for a stochastic gradient Nosé-Hoover thermostat with a general kinetic energy $K(p)$.

We first augment the Hamiltonian with ξ:

$$H(q, p, \xi) = U(q) + K(p) + \frac{d}{2}(\xi - D)^2.$$

We are now in the position to derive the SDE preserving the probability density

$\propto \exp(-H)$ by adopting the framework of [Ma et al., 2015] and defining:

$$H(\theta, p, \xi) = U(\theta) + K(p) + \frac{d}{2}(\xi - D)^2 \tag{3.14}$$

$$D(\theta, p, \xi) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & D \cdot I & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.15}$$

$$Q(\theta, p, \xi) = \begin{pmatrix} 0 & -I & 0 \\ I & 0 & \nabla K(p)/d \\ 0 & -\nabla K(p)^T/d & 0 \end{pmatrix}. \tag{3.16}$$

From (3.7) it follows that $\Gamma = (0\ 0\ -\Delta K(p)/d)^T$ and the dynamics becomes

$$\begin{pmatrix} d\theta \\ dp \\ d\xi \end{pmatrix} = \begin{pmatrix} \nabla K(p) \\ -\nabla \tilde{U} dt - \xi \nabla K(p) \\ \frac{1}{d}\left(\|\nabla K(p)\|^2 - \Delta K(p)\right) \end{pmatrix} dt + \begin{pmatrix} 0 & 0 & 0 \\ 0 & \sqrt{2D} & 0 \\ 0 & 0 & 0 \end{pmatrix} dW_t$$

where Δ is the Laplace operator defined as $\Delta K(p) = \sum_i \frac{\partial^2 K(p)}{\partial p_i^2}$. For the relativistic kinetic energy $K(p)$, we have that $\nabla_p K(p) = M^{-1}(p)p$ with $M(p) := m\left(\frac{p^T p}{m^2 c^2} + 1\right)^{\frac{1}{2}}$ a scalar and that $\Delta K(p) = tr\left(\frac{d}{dp}\left(\frac{1}{d} M^{-1}(p)p\right)\right)$. The Stochastic Gradient Nosé-Hoover Thermostat for relativistic HMC follows:

$$\begin{pmatrix} d\theta \\ dp \\ d\xi \end{pmatrix} = \begin{pmatrix} M^{-1}(p)p dt \\ -\nabla \tilde{U} - \xi M^{-1}(p)p \\ \frac{p^T p}{d}\left(M^{-2}(p) + c^{-2} M^{-3}(p)\right) - M^{-1}(p) \end{pmatrix} dt + \begin{pmatrix} 0 & 0 & 0 \\ 0 & \sqrt{2D} & 0 \\ 0 & 0 & 0 \end{pmatrix} dW_t.$$

3.4 Experiments

3.4.1 Toy examples

All the experimental results in this section are based on the separable versions (3.4) as they give superior results than the non-separable counterparts. We first explore the performances of the algorithms on a set of small examples including a two dimensional banana function (Banana) [Haario et al., 1999] with density $p(\mathbf{x}) \propto \exp\{-0.5(0.01x_1^2 + (x_2 + 0.1x_1^2 - 10)^2)\}$, and Gaussian mixture models

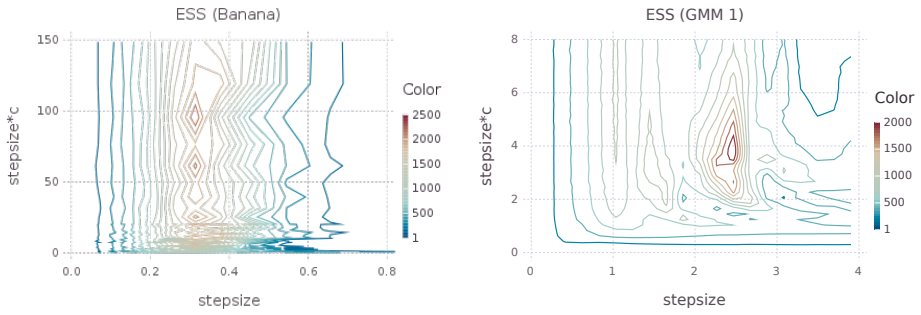


Figure 3.1: ESS contour plots of $\epsilon \times c$ versus ϵ for Banana (left) and GMM1 (right) datasets.

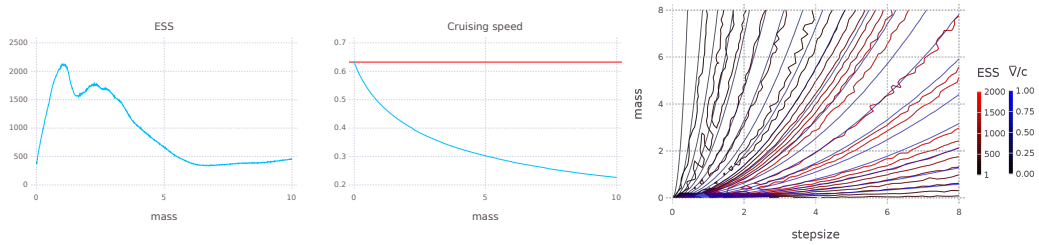


Figure 3.2: Varying m for GMM1. From left to right: ESS, cruising speed (the red horizontal line is c), and ESS and relative cruising speed $\bar{v}/c$ contour plots versus m and ϵ.

(GMM1, GMM2, GMM3) obtained by combining the three following Gaussian random variables with equal mixing proportions: $\mathcal{N}(-5, 1/\sigma^2)$, $\mathcal{N}(0, \sigma^2)$, $\mathcal{N}(5, 1/\sigma^2)$, where $\sigma^2 = 1, 0.5, 0.3$. When $\sigma^2 = 1$ the three Gaussians have the same variance and lower σ^2 means larger the discrepancies between their variances and thus a wider range of length scales and log density gradients. The density plots of the examples can be found in the top row of Figure 3.3 (the banana function results are by Xiaoyu Lu, co-author).

We start with an exploration of the behaviour of RHMC as the tuning parameters m, c and ϵ are varied. First we considered the effective sample sizes (ESS) of the algorithm on the Banana and GMM1 datasets. We varied both ϵ and $\epsilon \times c$ over a grid, and computed the average ESS, over 20 chains, each of length 10^4 for Banana, and over 100 chains of length 10^5 for GMM1. The ESS contour plots can be found in Figure 3.1, which suggests that ϵc and ϵ can be independently tuned. While ϵ controls the time discretisation of the continuous-time dynamics, ϵc controls the maximum change in the parameters at each leapfrog step. Next we varied

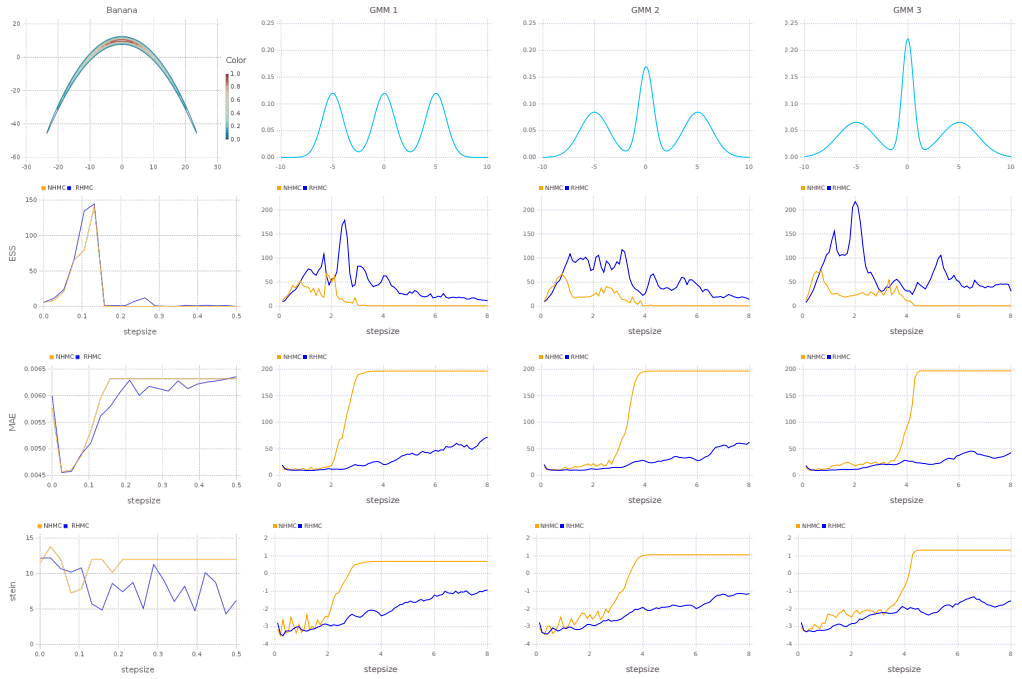


Figure 3.3: Left to right: Banana, GMM1, GMM2, GMM3 datasets. Top to bottom: density plot, ESS versus step size ϵ, MAE versus ϵ, log stein discrepancy versus ϵ.

the mass parameter m for GMM1, showing plots in Figure 3.2. As expected the ESS is optimised at an intermediate value of m, and the average "cruising speed" $\bar{v}$ decreases with m. In order to understand how to tune m, on the fourth panel we overlaid two contour plots: one for ESS and the other for $\bar{v}$. We see that the cruising speed $\bar{v}$ correlates much better with the ESS than m does, which suggests that m should be tuned via $\bar{v}$, e.g. by the user specifying a desired value for $\bar{v}$ and m being adapted to achieve the speed (noting that m and $\bar{v}$ have a monotonic relationship which makes for easy adaptation).

We next compare the performances of NHMC and RHMC for a wide range of step sizes, via the ESS (higher better), the *mean absolute error (MAE)* between the true probabilities and the histograms of the sample frequencies (lower better), and the log Stein discrepancy [Gorham and Mackey, 2015] which is a more accurate measure of sample quality (lower better). The reason being the Wasserstein distance can be bounded in terms of the Stein discrepancy thus accounting for bias and insufficient exploration of the target. The results can be found in rows 2-4 of Figure 3.3. It can be seen that RHMC achieves better performance and is strikingly more

robust to the step size ϵ than NHMC. As expected, this behaviour is particularly pronounced when the step size is large. Moreover, when the gradients of the target model span a large range of values (GMM2, GMM3), the improvements yielded by the relativistic variants are more pronounced. These results confirm that, since the speed of particles is bounded by c, RHMC is less sensitive to the presence of large gradients in the target density and more stable with respect to the choice of ϵ, allowing for a more efficient exploration of the target density.

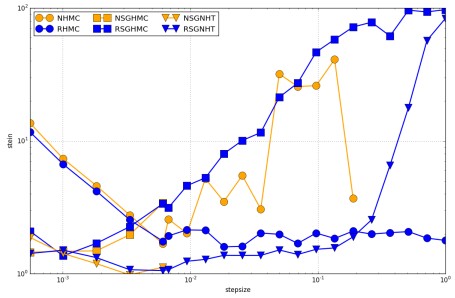


Figure 3.4: Stein discrepancy versus step size ϵ for logistic regression. NSGHMC and NSGNHT were unstable for $\epsilon > 6 \times 10^{-3}$ and thus their stein discrepancies were not plotted.

Next we compare both the Newtonian and relativistic variants of HMC and SGMCMC algorithms on a simulated 3-dimensional logistic regression example with 500 observations. For the stochastic versions of the algorithms, we use mini-batches of size 100. After a burn-in period of 1000 iterations, we calculated the Stein discrepancy for different ϵ while keeping the product $\epsilon \times c$ fixed. To make a fair comparison, we used 200 samples for NHMC and RHMC and 1000 samples for the SGMCMC algorithms. From Figure 3.4 (by Xiaoyu Lu, co-author), we see that the relativistic variants are significantly more robust than the Newtonian variants. The NHT algorithms were able to correct for stochastic gradient noise and performed better than SGHMC algorithms. Particularly, RSGNHT had lower Stein discrepancies than other algorithms for most values of ϵ.

3.4.2 Neural Networks

Turning to more complex models, we first considered a neural network with 50 hidden units and initialized its weights by the widely used Xavier initialization.

We used the Pima Indians dataset for binary classification (552 observations and 8 covariates) to compare the relativistic and the preconditioning approach. Indeed, these methods represent two different ways to normalise gradients so that the update sizes are reasonable for the local lengthscale of the target distribution. In particular we consider SGLD Adam, namely a preconditioned SGLD algorithm with an additional Adam-style debiasing of the preconditioner. Figure 3.5 compares the test-set accuracy of SGLD Adam with RSGD and RHMC, showing that the first is significantly outperformed by the relativistic algorithms. Due to Xavier initialization, all of the weights are small which causes small gradients, therefore the injected noise becomes very large due to the rescaling by the inverse of squared root of the average gradients, which makes SGLD Adam unstable. The histograms reveal that at the first iteration SGLD Adam causes the weights to become extremely large and this strongly compromises the performance of SGLD Adam, which takes a long time to recover. The relativistic framework represents therefore a much better approach to perform adaptation of the learning rates specific to each parameter.

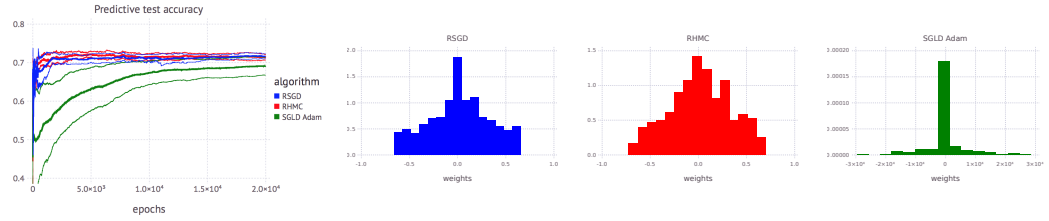


Figure 3.5: Comparison between RSGD, RHMC and SGLD Adam on the Pima Indians dataset using 50 hidden units. The histograms show the neural network weights at the first iteration.

We then apply our algorithms to the standard MNIST dataset, which consists 28×28 handwritten digital images from 10 classes with a training set of size $60,000$ and a test set of size $10,000$. We tested our optimization algorithm on a single layer with 100 hidden units and a multi-layer neural network with $500 * 300$ hidden units. In Figure 3.6 (by Xiaoyu Lu, co-author) a comparison with Adam and Santa [Chen et al., 2016] is displayed, their relation is discussed in more detail in Section 3.3.2. Note that, to ensure a fair comparison, we consider Santa SGD, namely a version of Santa that does not make use of symmetric splitting and simulated annealing. In other words, we adopt an Euler integration scheme

for all algorithms and consider the zero-temperature limit for Santa. It can be observed that our algorithm is competitive with Adam and is able to achieve a lower error rate, particularly with the 100 hidden units architecture. Moreover, RSGD performs significantly better than Santa SGD on all the considered architectures.

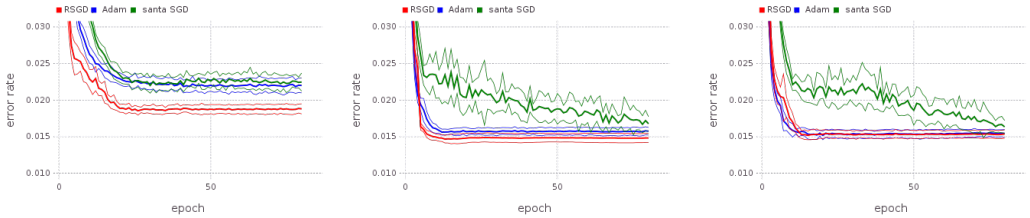


Figure 3.6: Comparison of error rate on MNIST dataset on the test set. From left to right: 100 hidden units; $500 * 300$ hidden units; $400 * 400$ hidden units.

3.5 Discussion

Our numerical experiments demonstrate that the relativistic algorithms discussed in this chapter are much more stable and robust to the choice of parameters and noise in stochastic gradients compared to the Newtonian counterparts. Moreover, we have a good understanding on how to choose the parameters c, m and ϵ. First the discretization parameter ϵ needs to be set, then we choose the maximal step $c \cdot \epsilon$ and in relation we choose the "cruising speed" $\frac{\bar{V}}{c}$ by picking m. The connection of our algorithms with popular stochastic optimizers such as Adam and RMSProp is novel and gives an interesting perspective to understand them.

Each of the proposed methodologies has scope for further research. The HMC version of the algorithm could be improved by employing some more advanced HMC methodology such as the NUTS version [Hoffman and Gelman, In press] and using partial moment refreshment instead of Adaptive Rejection Sampling [Neal, 2010]. The relativistic stochastic gradient descent seems to be very competitive with state of the art stochastic gradient methods for fitting neural networks. Further experimental work could investigate the properties of the relativistic algorithms to discern their ability to reduce noise in the stochastic gradient from their greater robustness to the discretization parameter. Additionally, better numerical integration schemes could be employed. We also anticipate a variety of

algorithms with different kinetic energies to be developed following our work. Last but not least, the strong simulation evidence should be complemented by more theoretical insights.

4

Multi-Task Adaptive Bayesian Linear Regression

Bayesian optimization (BO) is among the most successful applications of Bayesian nonparametric methods. BO is a model-based approach for gradient-free black-box function optimization, such as hyperparameter optimization, and is typically powered by a Gaussian process (GP). The algorithmic complexity of GPs, however, is cubic in the number of evaluations. Hence, GP-based BO cannot leverage large amounts of past or related function evaluations, for example, to warm-start related BO runs. In this chapter, we propose a multi-task adaptive Bayesian linear regression model, whose complexity is linear in the number of function evaluations. Specifically, we associate one Bayesian linear regression model to each black-box function optimization problem (or task), while we achieve transfer learning by coupling the models through a shared deep neural network. Experimental results show that the neural network learns a representation suitable for transfer learning between the black-box optimization problems and that BO runs can be accelerated when the target black-box function (e.g., validation loss) is learned together with other related signals (e.g., training loss).

4.1 Introduction

Bayesian optimization (BO) is a well-established methodology to optimize expensive black-box functions [Shahriari et al., 2016]. It relies on a probabilistic model of an unknown target function $f(x)$ we wish to optimize, which is repeatedly queried until one runs out of budget (e.g., time). The queries consist in evaluations of f at hyperparameter configurations $x^1, \ldots, x^n$ selected according to an explore-exploit trade-off criterion, such as the expected improvement [Mockus et al., 1978]. The hyperparameter configuration corresponding to the best query is then returned. One popular approach is to impose a Gaussian process (GP) prior over f and, in light of the observed queries $f(x^1), \ldots, f(x^n)$, to compute the posterior GP. The GP model maintains a posterior mean and a posterior variance function which are required to evaluate the explore-exploit criterion, or *acquisition function* [Mockus et al., 1978; Jones et al., 1998; Shahriari et al., 2016], for each new query of f.

Despite their flexibility and ability to calibrate the predictive uncertainty, standard GPs scale cubically with the number of observations [Rasmussen and Williams, 2006]. Hence, they cannot be applied in situations where f has been or can be queried a very large number of times. A possible alternative is to consider sparse GPs, which scale linearly in the number of observations and quadratically in the number of inducing points [Quinonero-Candela and Rasmussen, 2005; Titsias, 2009]. However, tractability requires the number of inducing points to be much smaller than the number of observations, resulting in a severe deterioration of the predictive performance [Wilson et al., 2015a].

In this chapter, we aim to warm-start BO in the context of hyperparameter optimization (HPO). Our goal is to learn across related black-box optimization problems by transferring information between them, thus leveraging data from previous BO runs. For example, we would like to warm-start the HPO of a given classifier when it is applied to a battery of reference data sets. Earlier attempts to tackle this specific transfer learning problem include the work by Bardenet et al. [2013] and Yogatama and Mann [2014].

To circumvent the scalability limitation of GPs, we propose to fall back to adaptive Bayesian linear regression (ABLR) [Bishop, 2006], which scales linearly

with the number of observations and cubically in the dimension of a learned basis function expansion, hence the name *adaptive*. Our first contribution is to extend ABLR to the multi-task setting. Here, a *task* denotes a black-box function optimization problem, associated with its own Bayesian linear regression surrogate. These models share an underlying feedforward neural network (NN) that learns a shared basis function expansion (or *representation*) from the HPO data. Our second contribution is to learn this representation *while* performing BO. This is made possible by integrating out the linear regression weights and to learn, both, the remaining parameters of each linear regression model and the NN weights by optimizing the resulting log marginal likelihood. To this end, our implementation leverages automatic differentiation operators recently contributed by Seeger et al. [2017] in MXNet [Chen et al., 2015].

The work is organized as follows. In Section 4.2, we relate our contributions to the state-of-the-art. In Section 4.3, we introduce our multi-task adaptive Bayesian linear regression model, detailing how to do inference in this model and discussing the computational properties. We also explain how the model and its attractive computational properties can be exploited for efficient transfer learning. Section 4.4 presents experiments on simulated and real data, reporting favorable comparisons with existing alternatives when leveraging data across auxiliary tasks and signals. We conclude with possible extensions in Section 4.5.

4.2 Background and related work

Consider the problem of optimizing a black-box function $f(x) : X \to \mathbb{R}$ over a convex set $X \subset \mathbb{R}$, namely a function whose analytic form and gradients are unavailable and that can only be queried through expensive and potentially noisy evaluations. For instance, suppose $f(x)$ is the test error associated to a deep neural network as a function of its hyper-parameters x (e.g., the number of layers, units, and type of activation functions). In this setting, each evaluation of $f(x)$ is typically very expensive as it requires training the neural network model.

BO is an efficient approach to find $x_\star = \text{argmin}_{x \in \mathcal{X}} \ f(x)$. The idea is to place a surrogate model over the target black-box, and update it sequentially

by querying $f(x)$ at new points that optimize an acquisition function, effectively trading off exploration and exploitation. Let M be a cheaper-to-evaluate surrogate model of $f(x)$, and C a set of evaluated candidates. The canonical BO loop iterates the following steps until some given budget, such as time, is exceeded:

1. Select a candidate $x_{\text{new}} \in X$ that optimizes a given acquisition function based on M and C.
2. Collect an evaluation y_{new} at $f(x_{\text{new}})$.
3. Update $C = C \cup \{(x_{\text{new}}, y_{\text{new}})\}$.
4. Update the surrogate model M based on C.
5. Update the budget.

A variety of models M have been used to describe the black-box function $f(x)$, with GPs being a common choice. In the next subsection, we review a set of alternative models that have been proposed to either overcome the scalability limits of GPs or extend BO to optimize multiple related black-box functions.

4.2.1 Related work

Our work is most closely related to DNGO [Snoek et al., 2015], where the same scalable ABLR model is used for BO. However, the authors only consider the single task setting with many evaluations. Apart from our focus on the multi-task extension of ABLR, we do not use their inconsistent two-step learning procedure. Namely, they first train the NN and the final *deterministic* linear regression layer together, leveraging standard deep learning software. Then, they discard the final layer, replacing it with a Bayesian linear regression model in order to drive BO. Our empirical results indicate that joint *Bayesian* learning of the ABLR parameters and the underlying NN parameters is beneficial, justifying the additional complexity in our implementation. Our procedure naturally extends to handling heterogeneous signals, and runs as fast as the two-step approach of Snoek et al. [2015]. More details on the relationship to DNGO are provided in Section 4.3.1 and 4.3.3.

Another related model is BOHAMIANN [Springenberg et al., 2016]. The authors propose using Bayesian NNs [Neal, 1996] to sample from the posterior over f and add task-specific embeddings to the NN inputs to handle the multi-task black-box function optimization setting. While allowing for a principled treatment of uncertainties, fully Bayesian NNs are computationally expensive and their training can be sensitive to the stochastic gradient MCMC procedure used to handle the hyperparameters. Our model allows for simpler inference and is more suitable for large scale deployment.

Sparse GPs for BO [McIntire et al., 2016] and multi-task GPs for BO [Swersky et al., 2013] have been respectively developed to scale up GPs and make them suitable for multi-task learning in the HPO setting. ABLR combines the strengths of these two approaches: it offers a simple alternative to the scaling issue in GPs and is applicable in the more general transfer learning setting.

A parallel line of research aims at exploiting knowledge coming from similar BO runs to speed up HPO. For instance, Feurer et al. [2015] warm-start HPO with the best known hyperparameters for the most similar black-box function, the similarity being defined in terms of the distance between the corresponding meta-data. ABLR learns a useful shared feature basis *even in the absence* of task meta-data. By leveraging this property and the resulting meta-data free formulation, we show that ABLR is able to learn task-specific embeddings as well. In addition, ABLR is able to draw information from *all* previous function evaluations, without limiting itself to the best solution from previous BO runs. This is similar in spirit to the work done by Swersky et al. [2013], Poloczek et al. [2016], and Poloczek et al. [2017], where the covariance matrix of a GP is designed to use the entire set of previous evaluations and capture black-box function similarities. ABLR makes it possible to fully embrace this idea by scaling up to orders of magnitude more observations than with a GP-based approach (linear rather than cubic scaling in the total number of evaluations).

A number of alternative models have been proposed specifically in the context of transfer learning for HPO. To this end, Schilling et al. [2015] model the interaction between data sets and optimal hyperparameters explicitly with a factorized

multilayer perceptron. Since this model is not able to represent uncertainties, an ensemble of 100 multilayer perceptrons is trained to obtain predictive means and simulate variances. A different approach is taken by Wistuba et al. [2016], where a two-stage surrogate model is developed: an independent GP is trained for each given data set, after which kernel regression combines the GPs into an overall surrogate model for BO. While the resulting model is able to exploit data set similarities, the cubic scaling makes GP-based approaches infeasible in the presence of a large number of evaluations. Finally, Golovin et al. [2017] consider transfer learning in the particular setting where a sequence order (e.g., time) is assumed across the BO runs; we do not require this assumption.

4.3 Multi-task Adaptive Bayesian Linear Regression

Consider T tasks, which consist in the target black-box functions $\{f_t(\cdot)\}_{t=1}^T$ we would like to optimize, and which are related in some way (e.g., the validation losses of a classification model learned on different data sets). We have evaluated $f_t(\cdot)$ N_t times, resulting in the data $D_t = \{(x_t^n, y_t^n)\}_{n=1}^{N_t}$, or $x_t \in \mathbb{R}^{N_t \times P}$ and $y_t \in \mathbb{R}^{N_t}$ in stacked form. Our joint model for the responses y_t consists of two parts. First, we use a shared feature map $\phi_z(x) : \mathbb{R}^P \mapsto \mathbb{R}^D$, parametrized by z. In our main use case, $\phi_z(x)$ is a feedforward NN with D output units, akin the model proposed by Bakker and Heskes [2003], and vector z collects all its weights and biases. We collect the features in matrices $\phi_t = \phi_z(x_t) = [\phi_z(x_t^n)]_n \in \mathbb{R}^{N_t \times D}$. Second, we employ separate Bayesian linear regression surrogates that share the feature map $\phi_z(x)$ to model the black-box functions:

$$P(y_t|w_t, z, \beta_t) = \mathcal{N}(\phi_t w_t, \beta_t^{-1} I_{N_t}),$$
$$P(w_t|\alpha_t) = \mathcal{N}(\mathbf{0}, \alpha_t^{-1} I_D),$$

where $\beta_t > 0$ and $\alpha_t > 0$ are precision (i.e., inverse variance) parameters. The model adapts to the scale and the noise level of the black-box function f_t via β_t and α_t, while the underlying NN parametrized by a shared vector z learns a representation to transfer information between the black-box functions. Importantly, the weights

w_t parametrizing the t^{th} Bayesian linear regression are treated as latent variables and integrated out, while the remaining parameters α_t, β_t and z are learned. The ABLR model can be seen as a NN whose final linear layers are Bayesian in the sense that their weights are integrated out rather than learned, or as a set of Bayesian linear regressions with a shared feature set learned by the NN. Note that Bayesian inference is analytically tractable and computationally efficient if restricted to the linear regression weights $\{w_t\}_{t=1}^T$.

4.3.1 Posterior Inference and Learning

Fixing the NN and precisions, the posterior distribution $P(w_t|D_t)$ over the linear regression weights are multivariate Gaussians, whose parameters can be computed analytically [Bishop, 2006]. Moreover, if $\phi_t^* = \phi_z(x_t^*)$ is a new input for task t and $f_t^* = w_t^\top \phi_t^*$ is the noise-free function value, the *predictive distribution* $P(f_t^*|x_t^*, D_t) = \int P(f_t^*|x_t^*, w_t)P(w_t|D_t)\, dw_t$ is Gaussian as well. Using Gaussian identities, it can be shown that $P(f_t^*|x_t^*, D_t) = \mathcal{N}(\mu_t(x_t^*), \sigma_t^2(x_t^*))$, where

$$\mu_t(x_t^*) = \frac{\beta_t}{\alpha_t}(\phi_t^*)^\top K_t^{-1}\phi_t^\top y_t = \frac{\beta_t}{\alpha_t}\mathbf{e}_t^\top L_t^{-1}\phi_t^*,$$
$$\sigma_t^2(x_t^*) = \frac{1}{\alpha_t}(\phi_t^*)^\top K_t^{-1}\phi_t^* + \frac{1}{\beta_t} = \frac{1}{\alpha_t}||L_t^{-1}\phi_t^*||^2 + \frac{1}{\beta_t}.$$

Here, $K_t = \beta_t\alpha_t^{-1}\phi_t^\top\phi_t + I_D$, and L_t is its Cholesky factor: $K_t = L_tL_t^\top$. Moreover, $\mathbf{e}_t = L_t^{-1}\phi_t^\top y_t$. The predictive mean and the predictive variance drive the BO procedure. Indeed, these quantities are required to compute the acquisition function, which is instrumental to identify the most promising hyperparameter configurations to evaluate next (see Shahriari et al. [2016] for a recent review and possible choices of acquisition functions).

BO with the multi-task ABLR models follows a similar procedure as in DNGO [Snoek et al., 2015]. A key difference is how the parameters $\{\alpha_t, \beta_t\}_{t=1}^T$ and z are learned. In DNGO, the NN weights are fit by using a *deterministic* final linear layer with weights w_1 (they consider $T = 1$ only), learning z and w_1 by stochastic gradient descent and discarding w_1 in subsequent BO rounds. By contrast, we make no difference between BO and learning, integrating out the latent weights w_t

in either case.

The learning criterion we minimize is the negative log *marginal likelihood* of the multi-task ABLR model:

$$\rho\left(z, \{\alpha_t, \beta_t\}_{t=1}^T\right) = -\sum_{t=1}^{T} \log P(y_t|z, \alpha_t, \beta_t), \tag{4.1}$$

where the marginal likelihood associated to task t is given by $P(y_t|z, \alpha_t, \beta_t) = \mathcal{N}(y_t|\mathbf{0}, \beta_t^{-1} I_{N_t} + \alpha_t^{-1}\phi_t\phi_t^\top)$. These quantities can also be expressed in terms of the Cholesky factor L_t of K_t. Alternatively, when $N_t < D$, we can work with the Cholesky factor of $I_{N_t} + \beta_t\alpha_t^{-1}\phi_t\phi_t^\top \in \mathbb{R}^{N_t \times N_t}$ instead. Hence, we can compute the learning criterion and its gradient in $\mathcal{O}(\sum_t \max(N_t, D)\min(N_t, D)^2)$.

4.3.2 Relationship with GPs

In our model, each ABLR could be seen as a GP using a shared linear kernel $\phi_z(x_1)^\top\phi_z(x_2)$, parameterized by z. Minimizing (4.1) is equivalent to learning these "kernel parameters" by empirical Bayes, as is routinely done for GPs [Rasmussen and Williams, 2006]. This viewpoint suggests that the uncertainty patterns learned by ABLR are, by construction, not as rich as the ones learned by a GP, implying a natural tradeoff between the lower computational cost and less informative representation of epistemic uncertainty. In our experiments, we find that the ability of ABLR to model uncertainties not to compromise performance in the transfer learning set-up. Therefore, we recommend using GPs for vanilla and low-data BO settings, while ABLR is desirable or necessary when several data points or a set of related tasks are available.

Furthermore, by integrating out the linear regression weights, we induce the learned feature map $\phi_z(x)$ to provide a good representation for covariance and dependencies, not just for good point predictions. By contrast, the DNGO learning procedure jointly learns features and weights of a linear regression model, hoping that the former give rise to a useful covariance function. The results we present in Section 4.4 provide evidence for the superiority of empirical Bayesian learning, at least in the multi-task context.

4.3.3 Computational Implications

Our learning procedure comes with an additional complexity compared to the two-step approach of DNGO, where the model is trained using standard deep NN software and stochastic gradient descent (SGD) on mini-batches. By contrast, while our learning criterion (4.1) decouples as a sum over tasks, it does not decouple over the observations within a task: all N_t observations for task t form a single batch. If the number of tasks T is moderate, our learning problem is best solved by batch optimization. In our experiments, L-BFGS [Byrd et al., 1995] worked well.

Since Bayesian learning and optimization are grounded in the same principle, we can re-train all model parameters as part of BO, whenever new evaluations become available for a task. We adopt this approach in all our experiments and noted that L-BFGS re-converges in few steps (often a single one) because parameters change little with each new observation. In situations with a large number of tasks, we could run BO on a task t by only updating (α_t, β_t), not retraining the NN or updating the other parameters $\{\beta_{t'}, \alpha_{t'}\}_{t' \neq t}$. Full model retraining could then be done offline.

Our learning criterion cannot be expressed in standard deep NN software. Namely, the evaluation of the Bayesian linear regression negative log marginal likelihood requires computations such as $K_t \mapsto L_t$ (Cholesky decomposition) and $(L_t, \mathbf{v}) \mapsto L_t^{-1}\mathbf{v}$ (backsubstitution). These have to be available as auto-grad operators and should run on, both, CPU and GPU, so they can be first-class citizens in a computation graph. We implemented ABLR in MXNet [Chen et al., 2015], where a range of linear algebra operators have recently been contributed [Seeger et al., 2017]. Given these operators, our implementation of ABLR is remarkable concise, and gradients required for model training and the minimization of the acquisition function are obtained automatically.

From a practical point of view, our approach has further advantages over DNGO. First, L-BFGS is simpler to use than SGD, as no parameters have to be tuned. This is all the more important in the context of BO, where our system has to work robustly on a wide range of problems without manual intervention. Second, we learn the parameters β_t and α_t separately for each task by empirical

Bayes [Mackay, 2003], while such parameters would have to be manually tuned in DNGO. The critical importance of this point is highlighted in Section 4.4.4.

4.3.4 Transfer Learning Settings

In our experiments in Section 4.4, we consider a range of different use cases of BO with the ABLR model.

The first use case we are interested in is HPO for a single machine learning model across different data sets. In this setting, a task consists in tuning the model on one of the data sets. Our goal is to warm-start HPO, so that a smaller number of evaluations are needed on a new data set, leveraging the logs of previous HPO runs. The simplest approach is to learn a common feature basis $\phi_z(x)$ across tasks, where each task is assigned to a separate marginal log likelihood term; see (4.1). If meta-features about the data set are further available [Feurer et al., 2015], we can collect them in a context vector c_t, and use a feature map $\phi_z(x, c_t)$ instead: the first part x of the input is variable, while the second part c_t is constant across data for a task.

Another use case is applying the ABLR model to a number of different signals (which play the role of tasks now). Here, we are interested in speeding up the optimization of one target function (e.g., validation loss), by leveraging a number of auxiliary signals (e.g., training cost, training loss considered at various epochs) which may come as a by-product or are cheaper to evaluate. Since these different signals can differ widely in scale and noise level, the automatic learning of the scale parameter α_t and the noise β_t is vitally important. Note that this set-up is different from a multi-objective scenario, such as the optimization of an average function over multiple tasks as described in Swersky et al. [2013]. Our set-up differs also from that of Poloczek et al. [2017] since our primary task is fixed beforehand and we do not seek to identify the best source of information at each round.

4.3.5 Online mode

It may be of interest in practical use cases to apply ABLR in an online setting, namely when not all tasks are available at the same time and one would like to

optimize a new task based on previous knowledge. One straightforward application of ABLR is to augment the pool of previous tasks with the new one, and train the model jointly. However, in some applications this is computationally prohibitive. An alternative approach is to train ABLR online by fixing the shared parameters and learning only the scale and weight parameter for a new task, in such a way to reduce the computational overhead and leave the full model retraining offline. Another possible direction would be to use task-specific meta-data to assess task similarities and therefore select a subset of the most relevant previous tasks to be used while training.

4.4 Experiments

The following subsections illustrate the benefits of multi-task ABLR in a variety of settings. In Sections 4.4.2 and 4.4.3, we evaluate its potential to transfer information between tasks defined by, respectively, synthetic data and OpenML data [Vanschoren et al., 2014]. In Section 4.4.4, we investigate the transfer learning ability of ABLR in presence of multiple heterogeneous signals. In either setting, our goal is to accelerate BO by leveraging data from the related tasks.

4.4.1 Experimental Set-up

We implemented multiple ABLR in GPyOpt [Gpy, 2016], with a backend in MXNet (see [Chen et al., 2015]), using recent linear algebra extensions [Seeger et al., 2017]. The NN that learns the feature map $\phi_z(x)$ is similar to the one used by Snoek et al. [2015]. It has three fully connected layers, each with 50 units and `tanh` activation function. Hence, 50 features are fed to the task-specific Bayesian linear regression models. We compare the NN set-up to random Fourier basis expansions [Rahimi and Recht, 2007], which have been successfully applied to BO [Hernández-Lobato et al., 2017; Jenatton et al., 2017]. Specifically, let $\mathbf{U} \in \mathbb{R}^{D\times P}$ and $\mathbf{b} \in \mathbb{R}^D$ be such that $\mathbf{U} \sim \mathcal{N}(\mathbf{0}, I)$ and $\{b_j\}_{j=1}^D \sim \mathcal{U}([0, 2\pi])$. For a vector x, the mapping is given by $\phi_z(x) = \sqrt{\frac{2}{D}} \cos\left(\frac{1}{\sigma}\mathbf{U}x + \mathbf{b}\right)$, where $\sigma \in \mathbb{R}^+$ is the bandwidth of the approximated radial basis function kernel. We will refer to this feature map as RKS ("random kitchen sink") in the remainder of the work. The RKS representation

has only a single parameter σ to optimize. We use the same L-BFGS code to learn it (see Section 4.3.3). We also compare ABLR-based BO to the standard GP-based BO. Here, we use GPyOpt. The GP has a Matérn-5/2 covariance kernel and automatic relevance determination hyperparameters, which are optimized by empirical Bayes [Rasmussen and Williams, 2006].

In the experiments, we will consider models with and without transfer learning. All models without transfer are initialized according to GPyOpt default settings, that is, with a set of five evaluations picked at random. The models with transfer are initialized with one random evaluation from the target task. All BO experiments use the expected improvement acquisition function [Mockus et al., 1978].

4.4.2 Transfer learning across parametrized quadratic functions

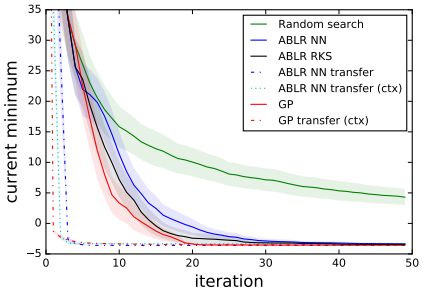


Figure 4.1: Transfer across parametrized quadratic functions. ABLR variants vs baselines. We plot the median over 30 leave-one-task-out runs and 10 independent repetitions, ± one standard error.

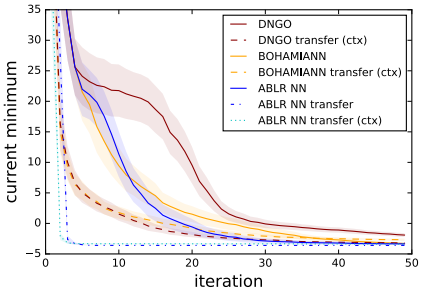


Figure 4.2: Transfer across parametrized quadratic functions. Comparison of NN-based methods. We plot the median over 30 leave-one-task-out runs and 10 independent repetitions, ± one standard error.

We first consider an artificial set-up with T tasks, each given by a quadratic function defined on $\mathbb{R}^3$:

$$f_t(x) = \frac{1}{2} a_{2,t} \|x\|_2^2 + a_{1,t} \mathbf{1}^\top x + a_{0,t},$$

where $(a_{2,t}, a_{1,t}, a_{0,t})$ belongs to $[0.1, 10]^3$. This triplet can be thought of as the context associated to each task t. We generated $T = 30$ different tasks by drawing $(a_{2,t}, a_{1,t}, a_{0,t})$ uniformly at random, and evaluated multi-task ABLR against baseline methods and NN-based methods in a leave-one-task-out fashion. Specifically,

we optimized each one of the 30 tasks after warm-starting the optimization with 10 observations drawn uniformly at random from each of the remaining 29 tasks. In other words, we have for each task $N_t = 10$ and warm-starting is therefore based on 290 evaluations. This set of observations is drawn once and taken the same for all other transfer learning methods. The results shown in Figure 4.1 and 4.2 are aggregates over 10 random repetitions of 30 leave-one-task-out runs.

In Figure 4.1, we compare single-task ABLR and standard GP driven HPO with their transfer learning counterparts. Transfer based on contextual information is denoted by `ctx`, using the context vector $c_t = [a_{2,t}, a_{1,t}, a_{0,t}]^T$. We perform transfer learning in standard GPs by stacking all observations together and augmenting the input space with the corresponding contextual information [Krause and Ong, 2011]. Note that `GP transfer` learning uses a single marginal likelihood criterion over the data from all tasks, while `ABLR NN transfer` learning models the data from different tasks as conditionally independent. HPO converges to the minimum much faster for all transfer learning variants (leveraging data from 29 related tasks) than for the single-task ones. The single-task ABLR based on the RKS representation with $D = 100$ performed comparably to the one based on the NN representation with $D = 50$. The dimension $D = 100$ was picked after we investigated the computation time of ABLR-based HPO with learned NN features ($D = 50$) and with RKS features ($D \in \{50, 100, 200\}$) and found that the running times were similar (see Figure 4.3). Figure 4.1 also shows that multi-task ABLR did not benefit much from the contextual information c_t.

We further benchmarked single-task and transfer ABLR against the state-of-the-art NN-based approaches DNGO [Snoek et al., 2015] and BOHAMIANN [Springenberg et al., 2016]. In contrast to GP-based methods, all these approaches scale linearly in the total number of evaluations. For their implementation we used the publicly available code.[1] We also used the recommended hyperparameters, which for BOHAMIANN were 2000 batches as burn-in followed by 2500 sampling steps. All NN architectures consisted of three fully connected layers, each with 50 units and `tanh` activation functions. Even though not considered in the original work, we extended these methods to the transfer learning case in the same way as the

[1] `https://github.com/automl/RoBO`

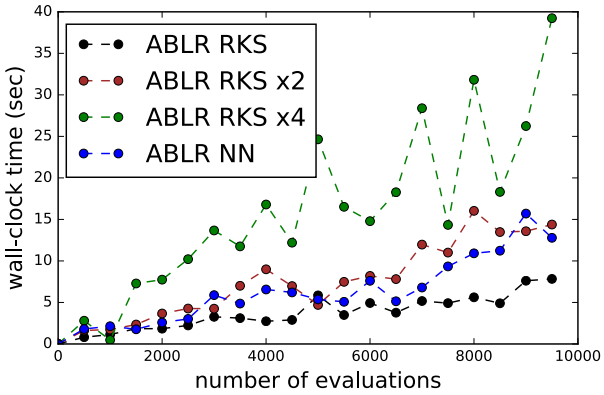


Figure 4.3: Comparing the scaling properties of ABLR-based HPO with NN learned features ($D = 50$) and RKS features ($D \in \{50, 100, 200\}$). The running time of NN-based ABLR includes the re-training of the NN with L-BFGS after each new evaluation.

GP baseline above, stacking all observations and augmenting the input space with contextual information. Different to multi-task ABLR, a single[2] marginal likelihood criterion is used over data from all tasks. Results are shown in Figure 4.2. The performance of single-task ABLR and BOHAMIANN is comparable (ABLR performs slightly better). DNGO and BOHAMIANN profit from transfer, yet less so than multi-task ABLR. Again, we note that the largest performance gain is realized *without* context input c_t. This suggests that multi-task ABLR learns a useful joint representation through its shared feature map and better exploit similarities across tasks.

Computational time

While the GP-based HPO with transfer slightly outperformed multi-task ABLR on the quadratic toy example, it does not scale to larger data sets, such as those considered in the next section. To make this more concrete, we measured the wall-clock time taken by HPO using GP and NN-based ABLR in a simple single-task setting. Our simulations showed that that GP-based HPO will not scale much beyond 2000 evaluations, which took approximately ten minutes, while ABLR-based HPO took only a few seconds. These results indicate that GP-based HPO is

[2]DNGO could also be used with separate marginal likelihood criteria per task, but this would need substantial changes of their code. Also, the parameters α_t and β_t would then have to be tuned for each task.

problematic when considering the multi-task setting.

Although all the considered NN-based algorithms are designed to scale linearly in N, with BOHAMIANN being slightly faster than DNGO [as observed in Springenberg et al., 2016], we found that our ABLR implementation requires much less computation time. More precisely, for the experiment in Figure 4.2, the average time $\pm$ one standard deviation per BO iteration over 300 repeated runs, on CPU, amounted to approximately 1.7 ± 0.10 seconds for single-task ABLR and approximately 28 ± 0.15 seconds for BOHAMIANN.

4.4.3 Transfer learning across OpenML data sets

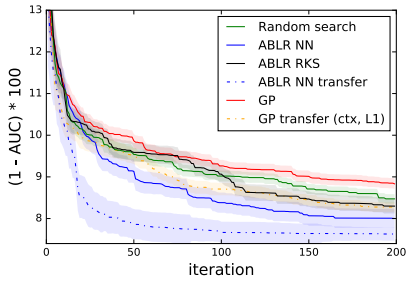


Figure 4.4: SVM transfer.

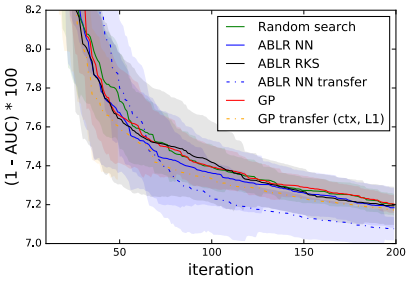


Figure 4.5: XGBoost transfer.

The ability to transfer knowledge across related tasks is particularly desirable in large-scale settings. Whenever runs from previous optimization-tasks are available, these can be used to warm-start and potentially speed up the current optimization. We consider the OpenML platform [Vanschoren et al., 2014], which contains a large number of evaluations for a wide range of machine learning algorithms (referred to as flows in OpenML) over different data sets. We focus on some of the most popular binary classification flows from OpenML, namely on a support vector machine (SVM, `flow_id` 5891) and extreme gradient boosting (XGBoost, `flow_id` 6767), and apply multi-task ABLR to optimize their hyperparameters. SVM comes with 4 and XGBoost with 10 hyperparameters. The parameters of the latter two exhibit conditional relationships, which we deal with by imputation [Lévesque et al., 2017]. We filtered the $T = 30$ most evaluated data sets for each `flow_id`, which amounts to $\sum_t N_t \approx 6.5 \times 10^4$ ($N_t \in [1.087, 3.250]$) for SVM

and $\sum_t N_t \approx 5.9 \times 10^5$ ($N_t \in [10.189, 48.615]$) for XGBoost. For these problems, the linear scaling of ABLR becomes almost mandatory. GP-based models cannot exploit all data even for a single task.

As previously, we apply a leave-one-task-out protocol, where each task stands for a data set. For the left-out task being optimized, say t_0, we use the surrogate modeling approach from Eggensperger et al. [2012]. We compare single-task variants `GP`, `ABLR RKS`, and `ABLR NN` which use evaluations of task t_0 only, with `ABLR NN transfer`, warm-started with the evaluations from the other 29 tasks, and `GP transfer (ctx, L1)`. In the latter approach, we warm-start the GP model with 300 randomly drawn data points from the closest task in terms of ℓ_1 distance between contextual features [similar to Feurer et al., 2015]. In all OpenML experiments, we chose four contextual features: number of data points, number of features, class imbalance, and a landmark feature based on Naive Bayes. We did not use those based on ensemble methods to avoid potential information leakage about the targets. Note that the multi-task ABLR variant is not provided with context features.

Results are reported in Figures 4.4 and 4.5, respectively for SVM and XGBoost. They indicate that evaluation data from other data sets helps to speed up convergence on a new task. In particular, `ABLR NN transfer` is able to leverage such data by way of learning an efficient shared set of features. We also tried to provide the context vector c_t as input to multi-task ABLR. However, this did not lead to robust improvements, and further explorations of this direction is left for future work. Also note that the performance of `ABLR NN` ($D = 50$) and `ABLR RKS` ($D = 100$) is comparable. The real benefit of learning features by empirical Bayes is only apparent in the multi-task scenario only.

Learning Task Embeddings

Instead of feeding context vectors c_t to the shared NN, we can try to *learn task embeddings*. This is of interest in a scenario where different ML methods are applied to a common (or at least overlapping) set of data sets. To this end, we use the NN form $\phi_z(x, c_t)$ from Section 4.3.4, together with $c_t = \mathbf{E}\boldsymbol{\delta}_t$. Here, $\mathbf{E} = [\mathbf{e}_t] \in \mathbb{R}^{k \times T}$

is a matrix of T embedding vectors, and $\boldsymbol{\delta}_t \in \{0,1\}^T$ is the one-hot vector for task $t \in \{1,\ldots,T\}$. The embedding matrix $\mathbf{E}$ becomes part of the overall parameter vector z, trained by empirical Bayes. In some preliminary experiments, we use ABLR NN to transfer across *two* dimensions. We operate in two stages. In these experiments we also consider a Random Forest (RF) model. First, for each of the methods RF, SVM, XGBoost, we learn NN features and task embeddings together over their respective $T = 30$ data sets. Second, we fix the task embeddings learned for one method and use them for another. In this second phase, we fix $\mathbf{E}$, but retrain the NN parameters. In our experiments, we fix the number k of contextual features to four (thus matching the dimension of the four OpenML meta-features used). The 30 data sets we transfer between for each method are not the same, but have substantial overlap. RF used the embeddings learned for XGBoost (18 common data sets), SVM used the same embeddings (15 data sets in common), and XGBoost used the embeddings learned for RF. The pairing was chosen to maximize the number of data sets in common. Results for this two-stage experiment are given in Figures 4.6, 4.7, and 4.8. The learned embeddings are competitive with the context vectors provided from human-crafted OpenML meta-features, but more work would be needed to establish the usefulness of learned or hand-designed context vectors.

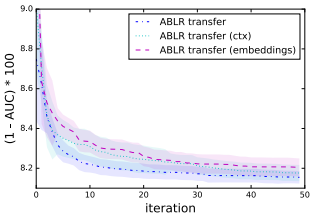


Figure 4.6: RF embeddings.

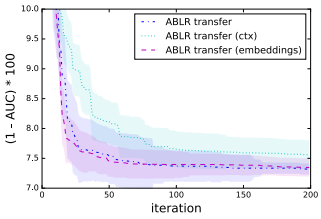


Figure 4.7: SVM embeddings.

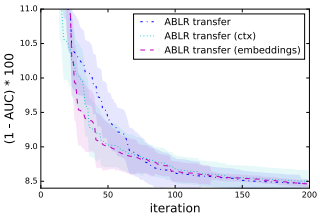


Figure 4.8: XGBoost embeddings.

4.4.4 Tuning feedforward neural networks from heterogeneous signals

In a final experiment, we consider tuning free parameters of feedforward NNs for binary classification.[3] As outlined in Section 4.3.4, we can use multi-task ABLR to simultaneously model T different signals. In our use case, we are interested in optimizing one of these signals (the test error), but model a range of auxiliary signals alongside (i.e., training error, training time, training error after e epochs). Put differently, we use the multi-task nature of ABLR to model T signals, learning a NN feature basis alongside a single HPO run. Importantly, the auxiliary signals come essentially for free, while most previous HPO algorithms do not seem to make use of them. Also note that different to the transfer learning settings above, we always evaluate all T signals together, at the same input points x. The fact that ABLR scales linearly in T, allows us to consider a large number of auxiliary signals (in contrast, multi-output GPs scales cubicallly in T). In our experiments, we tune the following four NN parameters: number of hidden layers in $\{1, \ldots, 4\}$, number of hidden units in $\{1, \ldots, 50\}$, ℓ_2 regularization constant in $\{2^{-6}, 2^{-5}, \ldots, 2^3\}$, and learning rate of Adam [Kingma and Ba, 2014a] in $\{2^{-6}, 2^{-5}, \ldots, 2^{-1}\}$.

Results are provided in Figure 4.9, averaged over 10 random repetitions and five data sets (`w8a, sonar, w1a, phishing, australian`) from LIBSVM [Chang and Lin, 2011]. The feedforward NN was trained for 200 iterations, each time on a batch of 200 samples. All variants model the test error as signal of interest (and target for HPO). `ABLR train loss (2)` also uses the final value of the training loss, `ABLR cost (2)` also models the CPU training time, while `ABLR cost + train loss (3)` models both. Finally, `ABLR cost + epochwise train loss (21)` uses the cost together with the training error collected every 10 iterations of the training. In the model names, the number in parentheses denotes the number T of signals modeled in ABLR. It is encouraging to see that adding auxiliary signals to an HPO run driven by ABLR NN speeds up convergence, the more so, the more signals are considered jointly. Note that this improvement comes from adding information which is available for free. We conjecture that adding auxiliary signals, related

[3] Here, we use one NN (providing the feature basis for ABLR) to tune the hyperparameters of a different family of NNs.

to the criterion of interest, helps in learning a useful feature basis by way of a feedfoward NN, *even if* only one of these signals is the target of HPO. It should also be noted that some of the signals we jointly model with ABLR NN, are of quite different scale (e.g., test error versus training time). The ability to learn different parameters $\{\alpha_t, \beta_t\}$ per signal automatically is vital here.

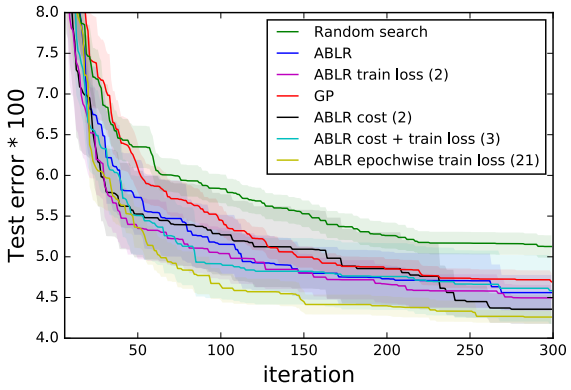


Figure 4.9: LIBSVM data, multiple signals.

4.5 Discussion

We introduced multi-task adaptive Bayesian linear regression (ABLR), a novel method for Bayesian optimization which scales linearly in the number of observations and is specifically designed for transfer learning in Bayesian optimization. Each task is modeled by a Bayesian linear regression layer on top of a shared feature map, which is learned jointly for all tasks with a deep neural network. Each Bayesian linear regression model comes with its own scale and noise parameters, which are learned with the neural network parameters by empirical Bayes. When leveraging the auto-grad operators for the Cholesky [Seeger et al., 2017], we found that training this model is as fast as the two-step heuristic recommended by Snoek et al. [2015].

We applied the multi-task ABLR model to two transfer learning problems for hyperparameter optimization (HPO). First, we investigated warm-starting HPO, considering synthetic black-box optimization problems and meta-learning problems from OpenML. We demonstrated that multi-task ABLR converges considerably

faster than Gaussian process methods or other NN-based approaches and scales to much larger sets of evaluations. We attribute the success of our method to its ability to learn useful shared feature map across multiple tasks, even in the absence of meta-data. We speculate that this is due to the specific loss structure, namely that it factorizes over the tasks, which can be exploited to learn useful task embeddings. Moreover, multi-task ABLR allows meta-data to be fed as context vectors to the shared neural network, allowing the features to become task-specific. This needs no further design decisions (such as task distance functions) or user tuning, but is part of the joint learning procedure. Second, we investigated multi-signal HPO for feedforward neural networks. We showed that multi-task ABLR is able to leverage a number of side-signals to speed up the optimization of the target signal (or task).

There are several extensions of this work that are of interest. The ABLR construct could be replaced by a GP based on deep kernels [Wilson et al., 2015b], which exhibits attractive scaling properties and does not suffer from the drawbacks of using sparse GPs. An empirical study would be required to evaluate if its multi-task extension warrants the additional implementation complexity. The Bayesian linear regression layers could also be complemented by Bayesian logistic regression layers in order to optimize binary signals or drive constrained HPO [Gelbart et al., 2014]. In a meta-learning context, we would have to further scale multi-task ABLR to a very large number of tasks, a regime where batch learning by L-BFGS has to be replaced by stochastic optimization at the level of tasks. Moreover, our joint Bayesian learning procedure for deep neural networks with a final Bayesian layer (which requires backpropagation through linear algebra operators such as Cholesky decomposition) may have applications to multi-task active learning or multi-label learning. Different to most other approximate Bayesian treatments of deep neural networks [Kingma and Welling, 2014; Rezende et al., 2014], we do not require random sampling or loosing variational bounding, but can fully leverage exact inference or tight approximation developed for generalized linear models as long as we only integrate out the weights of the last layer and learn the other parameters by empirical Bayes.

5

Bayesian Deep Learning for Population Genomic Data

In the previous chapter we introduced an adaptive Bayesian linear regression model that leverages deep learning to extract a useful representation from complex data. In this chapter, we pursue similar ideas to combine large-scale Bayesian inference with deep learning, and apply the latter to learn the first exchangeable feature representation for population genetic data.

Standard Bayesian inference in population genetics is challenging, as the likelihood term is often intractable. While this issue is commonly tackled by likelihood-free methods, these approaches rely on hand-crafted summary statistics of the data, whose design and selection is problematic in complex settings. In this chapter, we apply deep learning to work directly with genotype data. This is achieved by means of a novel Bayesian likelihood-free inference framework, where a permutation-invariant convolutional neural network learns the inverse functional relationship from the data to the posterior. We leverage access to scientific simulators to learn such likelihood-free function mappings, and establish a general framework for inference in a variety of simulation-based tasks. We demonstrate the power of our method on the recombination hotspot testing problem, outperforming the state-of-the-art.

5.1 Introduction

Statistical inference in complex population genetics models is challenging. These models are typically based on the coalescent [Kingman, 1982], a stochastic process describing the distribution over genealogies of a random sample of chromosomes from a large population. Despite the popularity of the coalescent, standard inference cannot be applied as likelihoods are often both analytically and computationally intractable.

This limitation can be overcome by likelihood-free methods such as Approximate Bayesian Computation (ABC) [Beaumont et al., 2002] and deep learning [Sheehan and Song, 2016]. These approaches leverage scientific simulators to draw samples from the generative model, and reduce population genetic data to a set of summary statistics prior to performing inference. In complex settings such as those in population genetics, designing and selecting suitable summary statistics is problematic and results are very sensitive to such choices.

Deep learning offers the possibility to avoid the need for hand-designed summary statistics in population genetic inference and work directly with genotype data. The goal of this work is to develop a scalable general-purpose inference framework for raw genetic data, without the need for summary statistics. We achieve this by designing a neural network which exploits the exchangeability in the underlying data to learn feature representations that can approximate the posterior accurately.

As a concrete example, we focus on the problem of recombination hotspot testing. Recombination is a biological process of fundamental importance, in which the reciprocal exchange of DNA during cell division creates new combinations of genetic variants. Experiments have shown that some species exhibit *recombination hotspots*, that is, short segments of the genome with high intensity recombination rates [Petes, 2001]. The task of recombination hotspot testing is to predict the location of recombination hotspots given genetic polymorphism data. Identifying recombination hotspots could provide invaluable insights into the biological process underlying recombination, and help geneticists map the alleles causing genetic diseases [Hey, 2004]. We demonstrate in experiments that we achieve state-of-the-art performance on the hotspot detection problem.

Our contributions focus on addressing major inferential challenges of complex population genetic inference. In Section 5.2 we review relevant lines of work in both the fields of machine learning and population genetics. In Section 5.3 we propose a scalable Bayesian likelihood-free inference framework for exchangeable data, which may be broadly applicable to many population genetic problems as well as more general simulator-based machine learning tasks. The application to population genetics is detailed in Section 5.4. In particular, we show how this allows for direct inference on the raw population genetic data, bypassing the need for *ad hoc* summary statistics. In Section 5.5 we run experiments to validate our method and demonstrate state-of-the-art performance in the hotspot detection problem.

5.2 Related Work

Likelihood-free methods like ABC have been widely employed in population genetics [Beaumont et al., 2002; Boitard et al., 2016; Wegmann et al., 2009; Sousa et al., 2009]. In ABC the parameter of interest is simulated from its prior distribution, and data are subsequently simulated from the generative model and reduced to a pre-chosen set of summary statistics. These statistics are compared to the summary statistics of the real data, and the simulated parameter is weighted according to the similarity of the statistics to derive an empirical estimate of the posterior distribution. However, choosing summary statistics for ABC is challenging because there is a trade-off between loss of sufficiency and computational tractability. In addition, there is no direct way to evaluate the accuracy of the approximation.

Other likelihood-free approaches have emerged from the machine learning community and have been applied to population genetics, such as support vector machines (SVMs) [Schrider and Kern, 2015; Pavlidis et al., 2010], single-layer neural networks [Blum and François, 2010], and deep learning [Sheehan and Song, 2016]. The connection between likelihood-free Bayesian inference and neural networks has also been studied previously by Jiang et al. [2015] and Papamakarios and Murray [2016]. An attractive property of these methods is that, unlike ABC, they can be applied to multiple datasets without repeating the training process. However, common practice in population genetics converts data into a set of summary statistics

before passing it through the machine learning models. Therefore, the success of such techniques is highly dependent on the ability to hand-craft informative statistics, which has to be repeated from scratch for each new problem setting.

The inferential accuracy and scalability of these methods can be improved by exploiting symmetries in the input data. Permutation-invariant models have been previously studied in machine learning for SVMs [Shivaswamy and Jebara, 2006] and, recently, gained a surge of interest in the deep learning literature. Recent work on designing architectures for exchangeable data include Ravanbakhsh et al. [2016], Guttenberg et al. [2016], and Zaheer et al. [2017], which exploit parameter sharing to encode invariances. To our knowledge, no prior work has been done on learning feature representations for exchangeable population genetic data.

We demonstrate these ideas on the problem of recombination hotspot testing. To this end, several methods have been developed [Fearnhead, 2006; Li et al., 2006; Wang and Rannala, 2009]. However, none of these are scalable to the whole genome, with the exception of `LDhot` [Auton et al., 2014; Wall and Stevison, 2016], so we limit our comparison to this latter method. `LDhot` relies on a composite likelihood, which can be seen as an approximate likelihood for summaries of the data. It can be computed only for a restricted set of models (i.e., an unstructured population with piecewise constant population size), is unable to capture dependencies beyond those summaries, and scales at least cubically with the number of DNA sequences. The method we propose in this chapter scales linearly in the number of sequences while using raw genetic data directly.

5.3 Methodology

In this section we propose a flexible framework to address the shortcomings of current likelihood-free methods. Although motivated by population genetics, we first lay out the ideas that generalize beyond this application. We describe the exchangeable representation in Section 5.3.1 and the training algorithm in Section 5.3.2, which are combined into a general likelihood-free inference framework in Section 5.3.3.

5.3.1 Feature Representation for Exchangeable Data

Population genetic datapoints $\mathbf{x}^{(i)}$ typically take the form of a binary matrix, where rows correspond to individuals and columns indicate the presence of a Single Nucleotide Polymorphism (SNP), namely a nucleotide variation at a given location of the DNA. For unstructured populations the order of individuals carries no information, hence the rows are exchangeable. More generally, given data $\mathbf{X} = (\mathbf{x}^{(1)}, \dots \mathbf{x}^{(N)})$ where $\mathbf{x}^{(i)} \in \mathbb{R}^{n \times d}$ and $\mathbf{x}^{(i)} := (x_1^{(i)}, \dots, x_n^{(i)}) \sim \mathbb{P}(\mathbf{x} \mid \theta^{(i)})$, we call $\mathbf{X}$ *exchangeably-structured* if, for every i,

$$\mathbb{P}\Big(x_1^{(i)}, \dots, x_n^{(i)} \mid \theta^{(i)}\Big) = \mathbb{P}\Big(x_{\sigma(1)}^{(i)}, \dots, x_{\sigma(n)}^{(i)} \mid \theta^{(i)}\Big),$$

for all permutations σ of the indices $\{1, \dots, n\}$ (definition by Jeffrey Chan, co-author).

To obtain an exchangeable feature representation of genotype data, we proceed as follows. Let $\Phi : \mathbb{R}^d \to \mathbb{R}^{d_1}$ be a feature mapping. We apply a symmetric function $g : \mathbb{R}^{n \times d_1} \to \mathbb{R}^{d_2}$ to the feature mapped datapoint to obtain $g\big(\Phi(x_1^{(i)}), \dots, \Phi(x_n^{(i)})\big)$, a feature representation of the exchangeably-structured data. This representation is general and can be adapted to various machine learning settings. For example, Φ could be some *a priori* fixed feature mapping (e.g. a kernel or summary statistics) in which case g should be chosen such that the resulting feature representation remains informative. More commonly, the mapping Φ needs to be learned (such as in kernel logistic regression or a deep neural network), hence we choose some fixed g such that subgradients can be backpropagated through g to Φ. Throughout the work, we choose to parameterize Φ with a neural network and choose g to be the element-wise max function, such that $g_j = \max\big(\Phi(x_1^{(i)})_j, \dots, \Phi(x_n^{(i)})_j\big)$. Empirical evidence on the performance of the max function is given in the experimental section. A variant of this representation is proposed by Ravanbakhsh et al. [2016] and Zaheer et al. [2017].

This embedding of exchangeably-structured data into a vector space is suitable for many tasks such as regression or clustering. We focus on inference in which the objective is to learn the function $f : \mathbb{R}^{n \times d} \to \mathcal{P}_\Theta$, where Θ is the space of all parameters θ and $\mathcal{P}$ is the space of all probability distributions on Θ.

Given our exchangeable feature representation, a function $h : \mathbb{R}^{d_2} \to \mathcal{P}_\Theta$ can be composed with a symmetric mapping. For simplicity, throughout the rest of the work we focus on binary classification where $\theta \in \{0, 1\}$, so that $\mathcal{P}_\theta$ can be parameterized by $\mathbb{P}(\theta = 1 \mid \mathbf{x}^{(i)}, \phi)$ where ϕ are nuisance parameters and h is parameterized as a neural network such that both h and Φ can be learned via backpropagation with a cross entropy loss. Specifically, we will apply this construction to infer the presence of recombination hotspots, indicated by the parameter θ. The posterior $\mathbb{P}(\theta = 1 \mid \mathbf{x}^{(i)}, \phi)$ is estimated by a soft max application so that the output is defined on $[0, 1]$.

This exchangeable representation has many advantages. While it could be argued that flexible machine learning models could learn the structured exchangeability of the data, encoding exchangeability explicitly allows for faster per-iteration computation and improved learning efficiency, since data augmentation for exchangeability scales as $O(n!)$.

Enforcing exchangeability implicitly reduces the size of the input space, allowing for much more tractable inference for large n. In addition, choices of g where d_2 is independent of n allows for a representation which is robust to differing number of exchangeable variables between train and test time. This property is particularly desirable to construct feature representations of fixed dimension even with missing data.

5.3.2 Simulation-on-the-fly

Supervised learning methods traditionally use a fixed training set and make multiple passes over the data until convergence. This training paradigm, however, can lead to poorly calibrated posteriors and overfitting. While the latter has largely been tackled by regularization methods and large datasets, the former has not been sufficiently addressed. Poorly calibrated posteriors are particularly an issue in scientific disciplines as scientists often demand methods with calibrated uncertainty estimates in order to measure the confidence behind new scientific discoveries (often leading to reliance on traditional methods with asymptotic guarantees such as MCMC).

When we have access to scientific simulators, the amount of training data available is limited only by the amount of compute time available for simulation, so we propose simulating each training datapoint on-the-fly such that there is exactly one epoch over the training data (i.e., no training point is passed through the neural network more than once). We refer to this as *simulation-on-the-fly*. We show that this approach guarantees properly calibrated posteriors and obviates the need for regularization techniques to address overfitting.

5.3.3 Likelihood-Free Inference Framework

With an exchangeable feature representation and an optimization procedure in hand, we can now combine these ingredients into an inference scheme. Let $\mathbf{x}$, θ, ϕ, and γ be the observed data, the latent parameter of interest, the nuisance parameters, and the prior hyperparameters, respectively. The latent parameter θ can be inferred by drawing samples from the prior distribution $\theta^{(i)}, \phi^{(i)} \sim \pi(\theta, \phi \mid \gamma)$ and from the density $\mathbf{x}^{(i)} \sim P(\mathbf{x} \mid \theta^{(i)}, \gamma, \phi^{(i)})$, while stochastic optimization under the simulation-on-the-fly paradigm fits $\tilde{f}_A(\mathbf{x}^{(i)})$ to $\theta^{(i)}$ in an online manner.

This Bayesian inference framework marginalizes over the uncertainty of the nuisance parameters. As neural networks have been empirically shown to interpolate well between examples, we recommend choosing a diffuse prior, which makes our trained model robust to model misspecification.

Another question about applying machine learning models for Bayesian inference is the calibration of the posteriors, since neural networks have been empirically shown to be overconfident in their predictions. Guo et al. [2017] showed that common deep learning practices cause neural networks to poorly represent uncertainty due to the noise inherent in the observations. We show in experiments that these calibration issues are a byproduct of the fixed training set regime but do not apply to simulation-on-the-fly. The softmax probabilities are calibrated for a correctly specified model under simulation-on-the-fly, since for a sufficiently expressive neural network the minimizer approximates the true posterior. However, under large model misspecification, softmax probabilities do not properly describe posteriors as they exhibit overconfidence in the presence of outliers. For recombina-

tion hotspot testing, we found that the summary statistics from the 1000 Genomes dataset [1000 Genomes Project Consortium, 2015] were similar to the summary statistics of the simulated data, so for simplicity we use the softmax probabilities as the posterior.

5.4 Population Genetics Application

The framework we established overcomes many challenges posed by population genetic inference. In this setting, each observation $\mathbf{x}$ is encoded as follows. Let $\mathbf{x}_S$ be the binary $n \times d$ allele matrix with 0 and 1 as the major and minor alleles respectively, where n is the number of individuals and d is the number of SNPs. Let $\mathbf{x}_D$ be the $n \times d$ matrix storing the distances between neighboring SNPs, so each row of $\mathbf{x}_D$ is identical and the rightmost distance is set to 0. Define $\mathbf{x}$ as the $n \times d \times 2$ tensor obtained by stacking $\mathbf{x}_S$ and $\mathbf{x}_D$. To improve the conditioning of the optimization problem, the distances are normalized such that they are on the order of $[0, 1]$. As mentioned in Section 5.3.1, this is an instance of exchangeably-structure data.

The standard generative model for such data is the coalescent, a stochastic process describing the distribution over genealogies relating samples from a population of individuals. The coalescent with recombination [Griffiths, 1981; Hudson, 1983] extends this model to describe the joint distribution of genealogies along the chromosome. The recombination rate between two DNA locations tunes the correlation between their corresponding genealogies. In order to take full advantage of parameter sharing, our chosen architecture is given by a convolutional neural network with tied weights for each row preceding the exchangeable layer, which is in turn followed by a fully connected neural network. We choose g as the element-wise max, and the architecture is depicted in Figure 5.1 (diagram by Jeffrey Chan, co-author).

5.4.1 Recombination Hotspot Testing

Recombination hotspots are short regions of the genome (≈ 2 kb in humans) with high recombination rate relative to the background recombination rate.

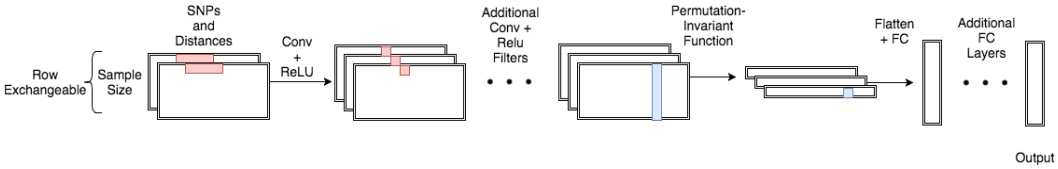


Figure 5.1: A cartoon schematic of the exchangeable architecture for population genetics.

Our prior is defined as follows (by Jeffrey Spence, co-author). We sample the hotspot indicator variable $h \sim \text{Bernoulli}(0.5)$ and the local recombination maps $\rho_w \sim \hat{P}(\rho_w \mid h)$ from the released fine-scale recombination maps of HapMap [Gibbs et al., 2003]. In addition, the demography is inferred via `SMC++` [Terhorst et al., 2017] and fixed in an empirical Bayes style throughout training for simplicity. The human mutation rate is fixed to that experimentally found in Kong et al. [2012]. Since `SMC++` is robust to changes in any small fixed window, inferring $\hat{\eta}$ from X has minimal dependence on ρ_w.

To test for recombination hotspots, first simulate a batch of h and ρ_w from the prior, and X_w from `msprime` [Kelleher et al., 2016]. Then, feed a batch of training examples into the network. Repeat until convergence or for a fixed number of iterations. At test time, slide along the genome to infer posteriors over h.

5.4.2 Recombination Hotspot Details

Recombination hotspots are short regions of the genome with high recombination rate relative to the background. In order to develop accurate methodology, a precise mathematical definition of a hotspot needs to be specified in accordance with the signatures of biological interest. We use the following definition (by Jeffrey Spence and Jeffrey Chan, co-authors).

Definition 1 (Recombination Hotspot) *Let a window over the genome be subdivided into three subwindows $w = (w_l, w_h, w_r)$ with physical distances α_l, α_h, and α_r, respectively, where $w_l, w_h, w_r \in \mathcal{G}$ where $\mathcal{G}$ is the space over all possible subwindows of the genome. Let a mean recombination map $R : \mathcal{G} \to \mathbb{R}_+$ be a function that maps from a subwindow of the genome to the mean recombination rate per base pair*

in the subwindow. A recombination hotspot for a given mean recombination map R is a window w which satisfies the following properties:

1. *Elevated local recombination rate:* $R(w_h) > k \cdot \max\big(R(w_l), R(w_r)\big)$
2. *Large absolute recombination rate:* $R(w_h) > k\tilde{r}$

where $\tilde{r}$ is the median (at a per base pair level) genome-wide recombination rate, and k is the relative hotspot intensity.

The first property is necessary to enforce the locality of hotspots and rule out large regions of high recombination rate, which are typically not considered hotspots by biologists. The second property rules out regions of minuscule background recombination rate in which sharp relative spikes in recombination still remain too small to be biologically interesting. The median is chosen here to be robust to the right skew of the distribution of recombination rates. Typically, for the human genome we use $\alpha_l = \alpha_r = 13$ kb, $\alpha_h = 2$ kb, and $k = 10$ based on experimental findings.

The most widely-used technique for recombination hotspot testing is `LDhot` as described in [Auton et al., 2014]. This is briefly summarized here, in the words of Jeffrey Spence and Jeffrey Chan, co-authors. The method performs a generalized composite likelihood ratio test using the two-locus composite likelihood based on [Hudson, 2001] and [McVean et al., 2004]. The composite two-locus likelihood approximates the joint likelihood of a window of SNPs w by a product of pairwise likelihoods

$$CL(\rho \mid \mathbf{x}) = \prod_{1 \leq |i-j| \leq z} L(\rho_{ij} \mid \mathbf{x}_{ij}),$$

where X_{ij} denotes the data restricted only to SNPs i and j, and ρ_{ij} denotes the recombination rate between those sites. Only SNPs within some distance, say $z = 50$, are considered.

Two-locus likelihoods are computed via an importance sampling scheme under a constant demography ($\eta = 1$) as in [McVean et al., 2004]. The likelihood ratio test uses a null model of a constant recombination rate and an alternative model of

a differing recombination rate in the center of the window under consideration:

$$\Lambda = -2 \log \left(\frac{\sup_{\rho_{\text{hot}}, \rho_{\text{bg}}} CL(\rho_{\text{hot}}, \rho_{\text{bg}} \mid X)}{\sup_{\rho_{\text{const}}} CL(\rho_{\text{const}} \mid X)} \right).$$

The two-locus likelihood can only be applied to populations with constant demography, constant mutation rate, and without natural selection. Furthermore, the two-locus likelihood is an uncalibrated approximation of the true joint likelihood. In addition, the experiments in Wall and Stevison [2016] and Auton et al. [2014] do not demonstrate the efficacy of `LDhot` against a realistic variable background recombination rate as its null hypothesis leads to a comparison against a biologically unrealistic flat background rate. In order to fairly compare our likelihood-free approach against the composite likelihood-based method in realistic human settings, we extended the `LDhot` methodology to apply to a piecewise constant demography using two-locus likelihoods computed by the software `LDpop` [Kamm et al., 2016]. Unlike the method described in Wall and Stevison [2016], our implementation of `LDhot` uses windows defined in terms of SNPs rather than physical distance in order to measure accuracy via ROC curves, since the likelihood ratio test is a function of number of SNPs. Note that computing the approximate two-locus likelihoods for a grid of recombination values is at least $O(n^3)$, which could be prohibitive for large sample sizes.

5.5 Experiments

In this section, we study the accuracy of our framework to test for recombination hotspots. As very few hotspots have been experimentally validated, we primarily evaluate our method on simulated data, with parameters set to match a human-like setting. The presence of ground truth allows us to benchmark our method and compare against `LDhot`. Unless otherwise specified, for all experiments we use the mutation rate, $\mu = 1.1 \times 10^{-8}$ per generation per nucleotide, convolution patch length of 5 SNPs, 32 and 64 convolution filters for the first two convolution layers, 128 hidden units for both fully connected layers, and 20-SNP length windows. The experiments comparing against `LDhot` used sample size $n = 64$ to construct

lookup tables for `LDhot` quickly. All other experiments use $n = 198$, matching the size of the CEU population (i.e., Utah Residents with Northern and Western European ancestry) in the 1000 Genomes dataset. All simulations were performed using `msprime` [Kelleher et al., 2016]. Gradient updates were performed using Adam [Kingma and Ba, 2014b] with learning rate $1 \times 10^{-3} \times 0.9^{b/10000}$, b being the batch count.

5.5.1 Evaluation of Exchangeable Representation

We compare the behavior of an explicitly exchangeable architecture to a nonexchangeable architecture that takes 2D convolutions with varying patch heights. The accuracy under human-like population genetic parameters with varying 2D patch heights is shown in Figure 5.2. Since each training point is simulated on-the-fly, data augmentation is performed implicitly in the nonexchangeable version without having to explicitly permute the rows of each training point. As expected, directly encoding the permutation invariance leads to more efficient training and higher accuracy while also benefiting from a faster per-batch computation time. Furthermore, the slight accuracy decrease when increasing the patch height confirms the difficulty of learning permutation invariance as n grows. Another advantage of exchangeable architectures is the robustness to the number of individuals at test time. As shown in Figure 5.3 (by Jeffrey Chan, co-author), the accuracy remains robust during test time for sample sizes roughly 0.5–4× the train sample size. Finally, besides the max function, we also explored the choice of several alternative permutation-invariant functions. Interestingly, the results in Figure 5.4 show that in this application the max tends to yield better test accuracy compared to the mean function. A generalization of the element-wise max function includes a range of sort functions, where the top $1, 2, \ldots, n$ elements are picked across individual representations. However, despite the extra computational cost, these did not lead to noticeable improvements, hence we used the max in all remaining experiments.

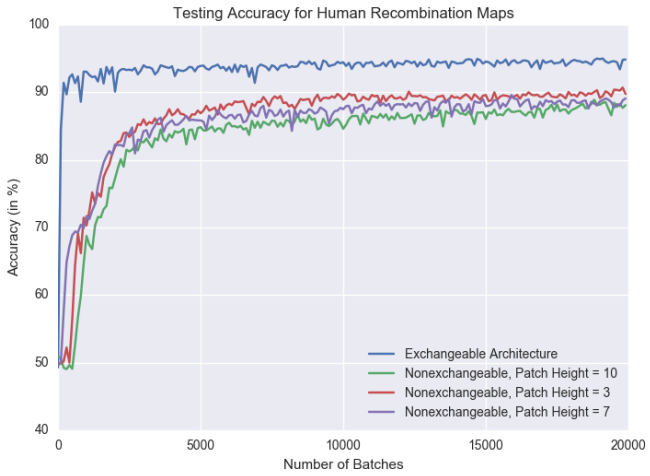


Figure 5.2: Accuracy comparison between exchangeable vs nonexchangeable architectures.

5.5.2 Evaluation of Simulation-on-the-fly

Next, we analyze the effect of simulation-on-the-fly in comparison to the standard fixed training set. A fixed training set size of 10000 was used and run for 20000 training batches and a test set of size 5000. For a network using simulation-on-the-fly, 20000 training batches were run and evaluated on the same test set. The weights were initialized with a fixed random seed in both settings with 20 replicates. Figure 5.5 (by Jeffrey Chan, co-author) shows that the fixed training set setting has both a higher bias and higher variance than simulation-on-the-fly. The bias can be attributed to the estimation error of a fixed training set in which the empirical risk surface is not a good approximation of the population risk surface. The variance can be attributed to an increase in the number of poor quality local optima in the fixed training set case.

We next investigated posterior calibration. This gives us a measure for whether there is any bias in the uncertainty estimates output by the neural network. We evaluated the calibration of simulation-on-the-fly against using a fixed training set of 10000 datapoints. The calibration curves were generated by evaluating 25000 datapoints at test time and binning their posteriors, computing the fraction of true labels for each bin. The results are given in Figure 5.6 (by Jeffrey Chan, co-author).

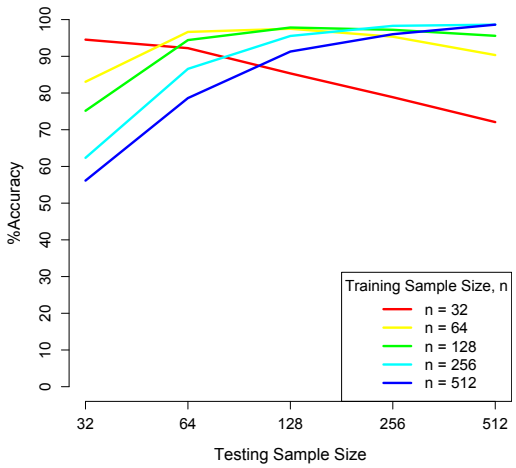


Figure 5.3: Performance of changing the number of individuals at test time for varying training sample sizes.

A perfectly calibrated curve is the dashed black line. The simulation-on-the-fly is much better calibrated with an increasing number of training examples leading to a more well-calibrated function. On the other hand, training with a fixed dataset leads to poor calibration.

5.5.3 Comparison to `LDhot`

We compared our method against `LDhot` in two settings: (i) sampling empirical recombination rates from the HapMap recombination map for CEU and YRI (i.e., Yoruba in Ibadan, Nigera) [Gibbs et al., 2003] to set the background recombination rate, and then using this background to simulate a flat recombination map with $10 - 100\times$ relative hotspot intensity, and (ii) sampling segments of the HapMap recombination map for CEU and YRI and classifying them as hotspot according to our definition, then simulating from the drawn variable map.

The ROC curves for both settings are shown in Figure 5.7 (by Jeffrey Chan, co-author). Under the bivariate empirical background prior regime where there is a flat background rate and flat hotspot, both methods performed quite well as shown on the top panel of Figure 5.7. We note that the slight performance decrease for YRI when using `LDhot` is likely due to hyperparameters that require tuning for each demography. This bivariate setting is the precise likelihood ratio

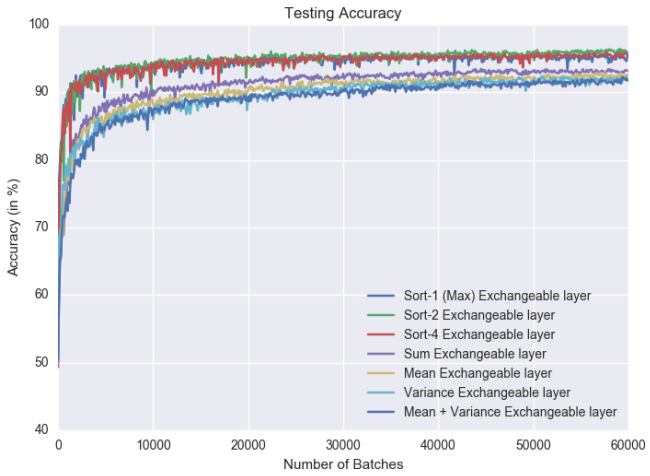


Figure 5.4: Performance for different exchangeable functions.

test for which `LDhot` tests. However, as flat background rates and hotspots are not realistic, we sample windows from the HapMap recombination map and label them according to a more suitable hotspot definition that ensures locality and rules out neglectable recombination spikes. The bottom panel of Figure 5.7 uses the same hotspot definition in the training and test regimes, and is strongly favorable towards the deep learning method. Under a sensible definition of recombination hotspots and realistic recombination maps, our method still performs well while `LDhot` performs almost randomly. We believe that the true performance of `LDhot` is somewhere between the first and second settings, with performance dominated by the deep learning method. Importantly, this improvement is achieved without access to any problem-specific summary statistics.

Our approach reached 90% accuracy in fewer than 2000 iterations, taking approximately 0.5 hours on a 64 core machine with the computational bottleneck due to the `msprime` simulation [Kelleher et al., 2016]. For `LDhot`, the two-locus lookup table for variable demography using the `LDpop` fast approximation [Kamm et al., 2016] took 9.5 hours on a 64 core machine (downsampling $n = 198$ from $N = 256$). The lookup table has a computational complexity of $O(N^3)$ while per-iteration training of the neural network scales as $O(n)$, allowing for much larger sample sizes.

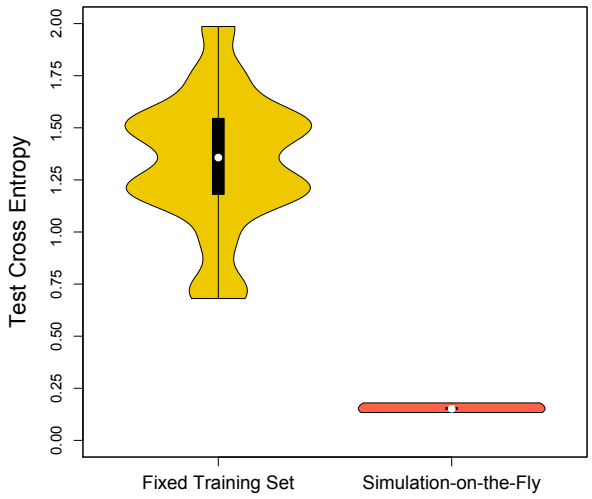


Figure 5.5: Comparison between the test cross entropy of a fixed training set of size 10000 and simulation-on-the-fly.

Finally, the hotspots called by the two methods in an example 500kb region are represented in Figure 5.8. The results show that the deep learning method tends to localize hotspots more precisely than LDhot, whose estimated hotspot regions tend to be overly large.

5.5.4 CEU data validation

We applied our model to detect hotspots on the real-data from the CEU population, chromosome 1. As little ground truth on hotspot presence is available, we aimed to validate our results by considering the R^2 values for all pairs of SNPs at a set of given distances. As SNPs that are further from each other and SNPs that span a true hotspot are less correlated, lower R^2 values are expected. Figure 5.9 shows the R^2 values computed by averaging across all pairs of SNPs with a set of given distances, separately for the SNPs that span a hotspot called by our method and the ones that do not span any called hotspots. The results show that pairs of SNPs spanning a called hotspot exhibit significantly lower correlation, confirming the validity of the hotspots called by the neural network.

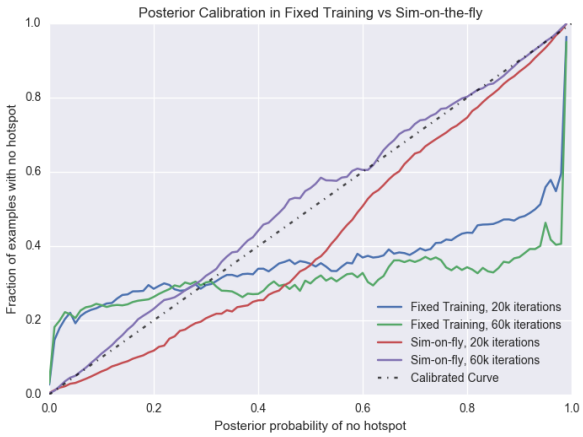


Figure 5.6: Posterior calibration. The black line is a perfectly calibrated curve. The red and purple lines are calibration curves for simulation-on-the-fly after 20000 and 60000 iterations, while the blue and green lines for a fixed training set of 10000 points, for 20000 and 60000 training iterations.

5.6 Discussion

We proposed the first likelihood-free inference method for exchangeable population genetic data that does not rely on handcrafted summary statistics. To achieve this, we designed a family of neural networks that learn an exchangeable representation of population genetic data, which is in turn mapped to the posterior distribution over the parameter of interest. Our simulation-on-the-fly training paradigm produced calibrated posterior estimates. State-of-the-art accuracy was demonstrated on the challenging problem of recombination hotspot testing.

The development and application of exchangeable neural networks to fully harness raw sequence data addresses an important challenge in applying machine learning to population genomics. The standard practice to reduce data to ad hoc summary statistics, which are then later plugged into a standard machine learning pipelines, is well recognized as a major shortcoming. Within the population genetic community, our method proves to be a major advance in likelihood-free inference in situations where ABC is too inaccurate. Several works have applied ABC to different contexts, and each one requires devising a new set of summary statistics. Our method can be extended in a black-box manner to these situations, which include inference on point clouds and quantifying evolutionary events.

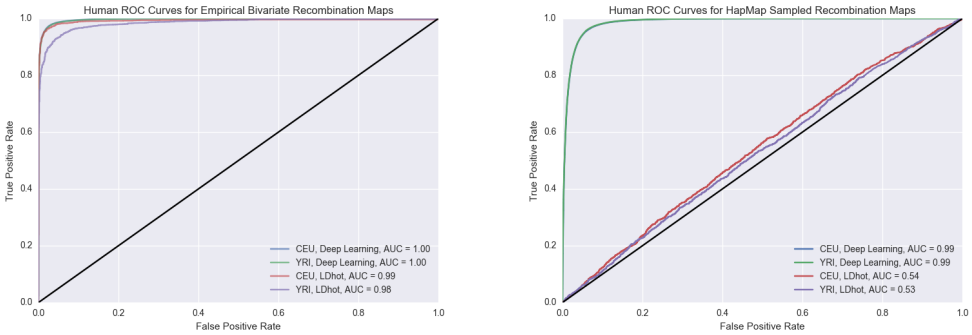


Figure 5.7: The black line represents a random classifier. (Top) ROC curve in the CEU and YRI setting for the deep learning and `LDhot` method. (Bottom) Windows of the HapMap recombination map drawn based on whether they matched up with our hotspot definition. The blue and green line coincide almost exactly.

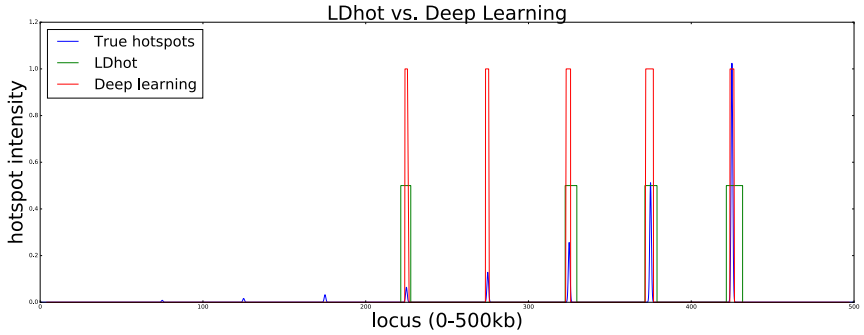


Figure 5.8: LDhot vs deep learning method. The bases of the green and red rectangles indicate the presence of an inferred hotspot (the heights are set for visualization purposes).

Quantifying uncertainty over a continuous parameter could be of interest in many other population genetic tasks, in which case softmax probabilities are inapplicable. Future work could adapt our method with ideas from the Bayesian neural networks literature to obtain posterior distributions over continuous parameters [Hernández-Lobato and Adams, 2015; Blundell et al., 2015; Gal and Ghahramani, 2016; Kingma et al., 2015].

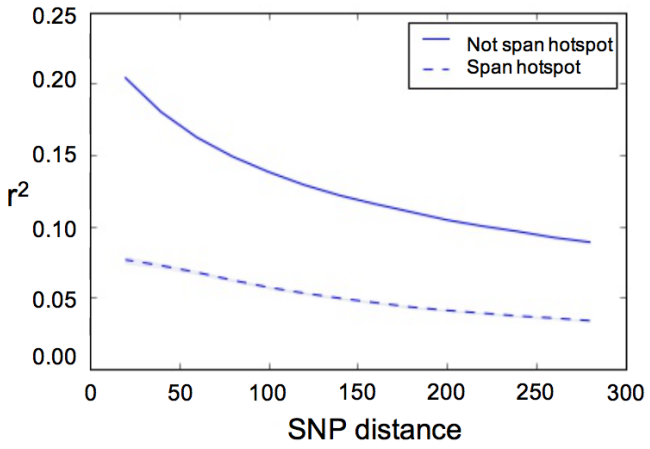


Figure 5.9: Comparison of the R^2 values averaged over all pairs of SNPs at the given distance, for pairs of SNP either spanning or not spanning a hotspot called by the deep learning method.

6

Conclusions

This thesis aims to contribute to the field of Bayesian machine learning by introducing a set of new models that can fully represent uncertainties while scaling to large and complex datasets. To this end, Chapter 2 introduced the *Wright-Fisher Indian Buffet process*, a novel Bayesian nonparametric model for time-dependent data. This model generalizes the beta and Indian Buffet process, leveraging a stochastic process from population genetics to reveal dynamic latent structure in the observed data. The model achieves improved performance over a static counterpart that treats all datapoints as exchangeable, and is able to provide insights on the evolution of latent features over time. The benefits of the model were demonstrated on the full collection of NIPS conference papers published up to 2015. As an example, we provided an interesting visualization of the popularity of competing approaches to machine learning over time.

In Chapter 3, we developed *Relativistic Monte Carlo*, a novel algorithm to sample from complex Bayesian models with large-scale data. The key insight was to modify the kinetic energy in Hamiltonian Monte Carlo to impose an upper bound on the speed of the parameter changes. In extensive experiments, the relativistic algorithms proved to be much more robust to the parameter choices. Furthermore, we developed stochastic gradient versions of these algorithms, which are particularly suitable to sample from complex and high-dimensional models such as Bayesian neural networks.

In Chapter 4, we proposed a simpler alternative to Bayesian neural networks,

i.e., *Adaptive Bayesian Linear regression* (ABLR). This model is able to provide full uncertainty estimates by means of a single Bayesian linear regression layer stack on top of a deep neural network. By sharing the neural network across different tasks, the model can transfer information across problems while adapting to the noise and scale level of each one of them by means of a task-specific Bayesian regression. In a Bayesian optimization application, this model was able to achieve state-of-the-art performance by scaling to millions of datapoints and using them to transfer knowledge across tasks.

Finally, Chapter 5 introduced a new likelihood-free framework for Bayesian learning in population genetics, leveraging deep neural networks to learn the first exchangeable representation of raw genomic data. As ABLR, this model combines the power of deep neural networks to learn flexible data representations with the ability of Bayesian inference to provide posterior distributions over the parameters of interest. In an application to the recombination hotspot detection task, the model outperformed the state-of-the-art.

6.1 Future work

A number of directions for future research are open. First, all the models we developed could be applied to different domains. Several applications beyond topic modeling and population genetics could be explored, especially in large-scale settings where a principled treatment of uncertainties is key. Second, the performance of Bayesian neural networks could be compared more in detail with ABLR, our simpler approach consisting of a deep neural network with a final Bayesian layer. Different to most other approximate Bayesian treatments of deep neural networks, ABLR does not require random sampling or loosing variational bounding. Instead, it can fully leverage exact inference or tight approximation developed for generalized linear models. We only integrate out the last layer weights and learn the others by empirical Bayes, hence inference is kept simple and scalable. Benchmarking this approach against fully Bayesian neural networks beyond the Bayesian optimization domain is an interesting avenue for future work. Third, the relativistic algorithms showed to be much more robust than existing stochastic gradient techniques to

sample from complex and high-dimensional models. The development of scalable and robust sampling algorithms could also facilitate the adoption of Bayesian neural network for real-world applications.

Most of the proposed models draw upon the strengths of deep learning and Bayesian modeling. While the Bayesian paradigm aims to establish principled and fully interpretable models, deep learning tools are typically black-boxes. Further work combining deep learning with Bayesian models could help build interpretable models with the flexible representations learned via deep learning. I hope that this thesis will contribute to a wider usage of Bayesian models in conjunction with deep learning to address the salient challenges of large-scale machine learning.

Bibliography

`GPyOpt`: A Bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.

1000 Genomes Project Consortium. A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015.

A. Ahmed and E. P. Xing. Timeline: A Dynamic Hierarchical Dirichlet Process Model for Recovering Birth/Death and Evolution of Topics in Text Stream. *UAI*, abs/1203.3463, 2012.

A. Amei and S. Sawyer. A time-dependent Poisson random field model for polymorphism within and between two related biological species. *Annals of Applied Probability*, 20(5):1663–1696, 2010.

A. Amei and S. Sawyer. Statistical inference of selection and divergence from a time-dependent poisson random field model. *PLoS ONE*, 7(4):e34413, 2012.

C. Andrieu and A. Doucet. Simulated annealing for maximum a posteriori parameter estimation of hidden markov models. *Information Theory, IEEE Transactions on*, 46(3):994–1004, 2000.

C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1–2):5–43, 2003.

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistal Society B*, 72:269–342, 2010.

A. Asuncion, M. Welling, P. Smyth, and Y. W. Teh. On Smoothing and Inference for Topic Models. *UAI*, pages 27–34, 2009.

A. Auton, S. Myers, and G. McVean. Identifying recombination hotspots using population genetic data. *arXiv: 1403.4264*, 2014.

B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *JMLR*, 4:83–89, 2003.

R. Bardenet, M. Brendel, B. Kégl, and M. Sebag. Collaborative hyperparameter tuning. In *ICML*, pages 199–207, 2013.

M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

A. Beskos, N. Pillai, G. O. Roberts, J. M. Sanz-Serna, and A. M. Stuart. Optimal tuning of hybrid Monte Carlo algorithm. *Bernoulli*, 19:1501–1534, 2013. doi: 10.3150/12-BEJ414.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer New York, 2006.

D. M. Blei and J. D. Lafferty. Dynamic topic models. *ICML*, pages 113–120, 2006.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *JMLR*, 3: 993–1022, 2003.

M. G. B. Blum and O. François. Non-linear regression models for Approximate Bayesian Computation. *Statistics and Computing*, 20(1):63–73, 2010.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. In *ICML*, volume 37, pages 1613–1622, Lille, France, 2015.

S. Boitard, W. Rodríguez, F. Jay, S. Mona, and F. Austerlitz. Inferring population size history from large samples of genome-wide molecular data-an approximate bayesian computation approach. *PLoS genetics*, 12(3):e1005877, 2016.

A. R. Boyko, S. H. Williamson, A. R. Indap, J. D. Degenhardt, R. D. Hernandez, K. E. Lohmueller, M. D. Adams, S. Schmidt, J. J. Sninsky, S. R. Sunyaev, T. J. White, R. Nielsen, A. G. Clark, and C. D. Bustamante. Assessing the evolutionary impact of amino acid mutations in the human genome. *PLoS Genet*, 4(5), 2008.

C. D. Bustamante, J. Wakeley, S. Sawyer, and D. L. Hartl. Directional Selection and the Site-Frequency Spectrum. *Genetics*, 159(4):1779–1788, 2001.

C. D. Bustamante, R. Nielsen, and D. L. Hartl. Maximum likelihood and Bayesian methods for estimating the distribution of selective effects among classes of mutations using DNA polymorphism data. *Theoretical Population Biology*, 63(2): 91–103, 2003. ISSN 0040-5809.

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5): 1190–1208, 1995.

M. P. Calvo. *Numerical hamiltonian problems*, volume 7. CRC Press, 1994.

B. Carpenter. Stan: A probabilistic programming language. *Journal of Statistical Software*, 2015.

C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

C. Chen, D. Carlson, Z. Gan, C. Li, and L. Carin. Bridging the gap between stochastic gradient MCMC and stochastic optimization. *AISTATS*, 2016.

T. Chen, E. Fox, and C. Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *ICML*, pages 1683–1691, 2014.

T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. In *Neural Information Processing Systems, Workshop on Machine Learning Systems*, 2015.

C. E. Dangerfield, D. Kay, S. MacNamara, and K. Burrage. A boundary preserving numerical algorithm for the Wright-Fisher model with mutation. *BIT Numerical Mathematics*, 52(2):283–304, 2012.

B. de Finetti. La prévision: ses lois logiques, ses sources subjectives. 7:1–68, 1937.

N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven. Bayesian Sampling Using Stochastic Gradient Thermostats. In *NIPS*, pages 3203–3211, 2014.

F. Doshi-Velez and Z. Ghahramani. Accelerated Sampling for the Indian Buffet Process. *ICML*, pages 273–280, 2009.

S. Duane, A.D. Kennedy, B.J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

A. Dubey, A. Hefny, S. Williamson, and E. P. Xing. A Nonparametric Mixture Model for Topic Modeling over Time. *SDM*, pages 530–538, 2013.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, July 2011. ISSN 1532-4435.

K. Eggensperger, F. Hutter, H. H. Hoos, and K. Leyton-brown. Efficient benchmarking of hyperparameter optimizers via surrogates background: hyperparameter optimization. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 1114–1120, 2012.

A. Einstein. On the Electrodynamics of Moving Bodies. *Annalen der Physik*, 17, 1905.

S. N. Ethier and T. G. Kurtz. *Markov processes: characterization and convergence*. Wiley series in probability and mathematical statistics. J. Wiley & Sons, 1986. ISBN 0-471-08186-8.

W. J. Ewens. *Mathematical Population Genetics*. Springer-Verlag, Berlin, 2004.

P. Fearnhead. SequenceLDhot: detecting recombination hotspots. *Bioinformatics*, 22:3061–3066, 2006.

M. Feurer, T. Springenberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *ICML*, volume 48, pages 1050–1059, New York, New York, USA, 2016.

M. A. Gelbart, J. Snoek, and R. P. Adams. Bayesian optimization with unknown constraints. Technical report, preprint arXiv:1403.5607, 2014.

S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6): 721–741, November 1984.

S. Geman and C. Hwang. Diffusions for global optimization. *SIAM Journal on Control and Optimization*, 24(5):1031–1043, 1986.

S. Gershman, P. I. Frazier, and D. M. Blei. Distance Dependent Infinite Latent Feature Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):334–345, 2015.

Z. Ghahramani, T. L. Griffiths, and P. Sollich. Bayesian nonparametric latent feature models. *Bayesian Statistics*, pages 201–225, 2007.

R. A. Gibbs, J. W. Belmont, P. Hardenbol, T. D. Willis, F. Yu, H. Yang, L.-Y. Ch'ang, W. Huang, B. Liu, Y. Shen, et al. The international hapmap project. *Nature*, 426(6968):789–796, 2003.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, March 2011. doi: 10.1111/j.1467-9868.2010.00765.x.

D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495, 2017.

J. Gorham and L. W. Mackey. Measuring Sample Quality with Stein's Method. In *NIPS*, pages 226–234, 2015.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola. A Kernel Method for the Two-Sample-Problem. *NIPS*, pages 513–520, 2006.

R. C. Griffiths. Neutral two-locus multiple allele models with recombination. *Theoretical Population Biology*, 19(2):169–186, 1981.

R. C. Griffiths. The frequency spectrum of a mutation, and its age, in a general diffusion model. *Theoretical Population Biology*, 64(2):241–251, 2003.

T. L. Griffiths and Z. Ghahramani. The Indian Buffet Process: An Introduction and Review. *JMLR*, 12:1185–1224, 2011.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *PNAS*, 101:5228–5235, 2004.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. *arXiv:1706.04599*, 2017.

R. N. Gutenkunst, R. D. Hernandez, S. H. Williamson, and C. D. Bustamante. Inferring the Joint Demographic History of Multiple Populations from Multidimensional SNP Frequency Data. *PLoS Genet*, 5(10), 2009.

N. Guttenberg, N. Virgo, O. Witkowski, H. Aoki, and R. Kanai. Permutation-equivariant neural networks applied to dynamics prediction. *arXiv:1612.04530*, 2016.

H. Haario, E. Saksman, and J. Tamminen. Adaptive Proposal Distribution for Random Walk Metropolis Algorithm. *Computational Statistics*, 14(3):375–396, 1999.

D. L. Hartl, E. N. Moriyama, and S. A. Sawyer. Selection intensity for codon bias. *Genetics*, pages 227–234, 1994.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

J. M. Hernández-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*, pages 1861–1869, 2015.

J. M. Hernández-Lobato, J. Requeima, E. O. Pyzer-Knapp, and A. Aspuru-Guzik. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *ICML*, 2017.

J. Hey. What's so hot about recombination hotspots? *PLoS Biol*, 2(6):e190, 2004.

M. D. Hoffman and A. Gelman. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *JMLR*, In press.

R. R. Hudson. Properties of a neutral allele model with intragenic recombination. *Theoretical population biology*, 23(2):183–201, 1983.

R. R. Hudson. Two-locus sampling distributions and their application. *Genetics*, 159(4):1805–1817, 2001.

R. Jenatton, C. Archambeau, J. Gonzales, and M. Seeger. Bayesian optimization with tree-structured dependencies. In *ICML*, 2017.

P. A. Jenkins and D. Spanò. Exact simulation of the Wright-Fisher diffusion. *Annals of Applied Probability*, 2017. To appear.

B. Jiang, T.-y. Wu, C. Zheng, and W.H. Wong. Learning summary statistic for approximate Bayesian computation via deep neural network. *arXiv:1510.02175*, 2015.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

J. A. Kamm, J. P. Spence, J. Chan, and Y. S. Song. Two-locus likelihoods under variable population size and fine-scale recombination rate estimation. *Genetics*, 203(3):1381–1399, 2016.

S. Karlin and H. M. Taylor. *A Second Course in Stochastic Processes*. Academic Press, 1981.

J. Kelleher, A. M. Etheridge, and G. McVean. Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLoS computational biology*, 12(5): e1004842, 2016.

D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. dec 2014a.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014b.

D. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Representation Learning*, 2014.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014c.

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *NIPS*, pages 2575–2583, 2015.

J. F. C. Kingman. Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78, 1967.

J. F. C. Kingman. The coalescent. *Stochastic processes and their applications*, 13 (3):235–248, 1982.

A. Kong, M. L. Frigge, G. Masson, S. Besenbacher, P. Sulem, G. Magnusson, S. A. Gudjonsson, A. Sigurdsson, A. Jonasdottir, A. Jonasdottir, et al. Rate of de novo mutations and the importance of father's age to disease risk. *Nature*, 488(7412): 471–475, 2012.

A. Krause and C. S. Ong. Contextual gaussian process bandit optimization. In *NIPS*, pages 2447–2455, 2011.

N. M. Laurendeau. *Statistical Thermodynamics: Fundamentals and Applications*. Cambridge University Press, 2005.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, 1989. ISSN 0899-7667.

Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

B. Leimkuhler and S. Reich. A metropolis adjusted nosé-hoover thermostat. *ESAIM: Mathematical Modelling and Numerical Analysis*, 43:743–755, 7 2009. ISSN 1290-3841. doi: 10.1051/m2an/2009023.

B. Leimkuhler and X. Shang. Adaptive Thermostats for Noisy Gradient Systems. *SIAM Journal on Scientific Computing*, 38(2):A712–A736, 2016. ISSN 1064-8275. doi: 10.1137/15M102318X.

J. Lévesque, A. Durand, C. Gagné, and R. Sabourin. Bayesian optimization for conditional hyperparameter spaces. In *International Joint Conference on Neural Networks (IJCNN)*, 2017.

C. Li, C. Chen, D. Carlson, and L. Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. *arXiv preprint arXiv:1512.07666*, 2015.

J. Li, M. Q. Zhang, and X. Zhang. A new method for detecting human recombination hotspots and its applications to the hapmap encode data. *The American Journal of Human Genetics*, 79(4):628–639, 2006.

X. Lu, V. Perrone, L. Hasenclever, Y. W. Teh, and S. J. Vollmer. Relativistic Monte Carlo. *AISTATS*, 54:1236–1245, 2017.

Y. Ma, T. Chen, and E. B. Fox. A Complete Recipe for Stochastic Gradient MCMC. jun 2015.

D. J. C. Mackay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

M. McIntire, D. Ratner, and S. Ermon. Sparse Gaussian processes for Bayesian optimization. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2016.

G. A. T. McVean, S. R. Myers, S. Hunt, P. Deloukas, D. R. Bentley, and P. Donnelly. The fine-scale structure of recombination rate variation in the human genome. *Science*, 304:581–584, 2004.

K. T. Miller, T. L. Griffiths, and M. I. Jordan. The Phylogenetic Indian Buffet Process: A Non-Exchangeable Nonparametric Prior for Latent Features. *UAI*, abs/1206.3279:403–410, 2012.

J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129):2, 1978.

R. M. Neal. *Bayesian Learning for Neural Networks*. Number 118 in Lecture Notes in Statistics. Springer, 1996.

R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.

Y. Ogata. On Lewis' simulation method for point processes. *IEEE Transactions on Information Theory*, 27(1):23–30, 1981.

G. Papamakarios and I. Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. *arXiv:1605.06376*, 2016.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

S. Patterson and Y. W. Teh. Stochastic gradient riemannian langevin dynamics on the probability simplex. In *NIPS*, pages 3102–3110, 2013.

P. Pavlidis, J. D. Jensen, and W. Stephan. Searching for footprints of positive selection in whole-genome snp data from nonequilibrium populations. *Genetics*, 185(3):907–922, 2010.

V. Perrone, P. A. Jenkins, D. Spanò, and Y. W. Teh. Poisson Random Fields for Dynamic Feature Models. *JMLR*, 18(127):1–45, 2017.

T. D. Petes. Meiotic recombination hot spots and cold spots. *Nature Reviews Genetics*, 2(5):360–369, 2001.

M. Poloczek, J. Wang, and P. I. Frazier. Warm starting bayesian optimization. In *Winter Simulation Conference, WSC 2016, Washington, DC, USA, December 11-14, 2016*, pages 770–781, 2016.

M. Poloczek, J. Wang, and P. Frazier. Multi-information source optimization. In *NIPS 30*, pages 4291–4301, 2017.

J. Quinonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *JMLR*, 6:1939–1959, 2005.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, pages 1177–1184, 2007.

V. Rao and Y. W. Teh. Spatial Normalized Gamma Processes. *NIPS*, pages 1554–1562, 2009.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

S. Ravanbakhsh, J. Schneider, and B. Poczos. Deep learning with sets and point clouds. *arXiv:1611.04500*, 2016.

D. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and variational inference in deep latent Gaussian models. In *ICML*, 2014.

S. A. Sawyer and D. L. Hartl. Population genetics of polymorphism and divergence. *Genetics*, 132(4):1161–1176, 1992.

N. Schilling, M. Wistuba, L. Drumond, and L. Schmidt-Thieme. Hyperparameter Optimization with Factorized Multilayer Perceptrons. *Proceedings of the European Conference on Machine Learning (ECML)*, pages 87–103, 2015.

D. R. Schrider and A. D. Kern. Inferring selective constraint from population genomic data suggests recent regulatory turnover in the human brain. *Genome biology and evolution*, 7(12):3511–3528, 2015.

M. Seeger, A. Hetzel, Z. Dai, and N. D. Lawrence. Auto-differentiating linear algebra. Technical report, preprint arXiv:1710.08717, 2017.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

X. Shang, Z. Zhu, B. Leimkuhler, and A. J. Storkey. Covariance-Controlled Adaptive Langevin Thermostat for Large-Scale Bayesian Sampling. In *NIPS*, pages 37–45, 2015.

S. Sheehan and Y. S. Song. Deep learning for population genetic inference. *PLoS Computational Biology*, 12(3):e1004845, 2016.

P. K. Shivaswamy and T. Jebara. Permutation invariant svms. In *ICML*, pages 817–824, 2006.

U. Şimşekli, R. Badeau, A. T. Cemgil, and G. Richard. Stochastic Quasi-Newton Langevin Monte Carlo. *ICML*, 2016.

J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, Mr. Prabhat, and R. Adams. Scalable Bayesian optimization using deep neural networks. In *ICML*, pages 2171–2180, 2015.

V. C. Sousa, M. Fritz, M. A. Beaumont, and L. Chikhi. Approximate Bayesian computation without summary statistics: The case of admixture. *Genetics*, 181 (4):1507–1519, 2009.

J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust Bayesian neural networks. In *NIPS*, pages 4134–4142, 2016.

K. Swersky, J. Snoek, and R. P. Adams. Multi-task Bayesian optimization. In *NIPS*, pages 2004–2012, 2013.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

Y. W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking Construction for the Indian Buffet Process. *AISTATS*, 11:556–563, 2007.

Y. W. Teh, L. Hasenclever, T. Lienart, S. Vollmer, S. Webb, B. Lakshminarayanan, and C. Blundell. Distributed bayesian learning with stochastic natural-gradient expectation propagation and the posterior server. *CoRR*, abs/1512.09327, 2015.

J. Terhorst, J. A. Kamm, and Y. S. Song. Robust and scalable inference of population history from hundreds of unphased whole genomes. *Nature genetics*, 49(2): 303–309, 2017.

R. Thibaux and M. I. Jordan. Hierarchical beta processes and the Indian buffet process. *AISTATS*, 2:564–571, 2007.

T. Tieleman and G. Hinton. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude, 2012. COURSERA: Neural Networks for Machine Learning.

M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *AISTATS*, pages 567–574, 2009.

J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

S. G. Walker. Sampling the Dirichlet mixture model with slices. *Communications in Statistics - Simulation and Computation*, 36(1):45–54, 2007.

J. D. Wall and L. S. Stevison. Detecting recombination hotspots from patterns of linkage disequilibrium. *G3: Genes, Genomes, Genetics*, 2016.

Y. Wang and B. Rannala. Population genomic inference of recombination rates and hotspots. *Proceedings of the National Academy of Sciences*, 106(15):6215–6219, 2009.

D. Wegmann, C. Leuenberger, and L. Excoffier. Efficient Approximate Bayesian Computation coupled with Markov chain Monte Carlo without likelihood. *Genetics*, 182(4):1207–1218, 2009.

M. Welling and Y.W. Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *ICML*, pages 681–688. Omnipress, 2011.

S. Williamson, C. Wang, K. Heller, and D. Blei. Focused Topic Models. *NIPS Workshop on Applications for Topic Models: Text and Beyond*, pages 1–4, 2009.

S. Williamson, P. Orbanz, and Z. Ghahramani. Dependent Indian Buffet Processes. *AISTATS*, pages 924–931, 2010a.

S. Williamson, C. Wang, K. A. Heller, and D. M. Blei. The IBP Compound Dirichlet Process and its Application to Focused Topic Modeling. *ICML*, pages 1151–1158, 2010b.

S. H. Williamson, R. Hernandez, A. Fledel-Alon, L. Zhu, R. Nielsen, and C. D. Bustamante. Simultaneous inference of selection and population growth from patterns of variation in the human genome. *PNAS*, 102(22):7882–7887, 2005.

A. G. Wilson, C. Dann, and H. Nickisch. Thoughts on massively scalable gaussian processes. Technical report, preprint arXiv:1511.01870, 2015a.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. *arXiv preprint arXiv:1511.02222*, 2015b.

M. Wistuba, N. Schilling, and L. Schmidt-Thieme. Two-Stage Transfer Surrogate Model for Automatic Hyperparameter Optimization. *Proceedings of the European Conference on Machine Learning (ECML)*, pages 199–214, 2016.

T. Xifara, C. Sherlock, S. Livingstone, S. Byrne, and M. Girolami. Langevin diffusions and the metropolis-adjusted langevin algorithm. *Statistics & Probability Letters*, 91(C):14–19, 2014.

D. Yogatama and G. Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *AISTATS*, pages 1077–1085, 2014.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep sets. *NIPS*, 2017.

M. Zhou, H. Yang, G. Sapiro, D. B. Dunson, and L. Carin. Dependent Hierarchical Beta Process for Image Interpolation and Denoising. *AISTATS*, 15:883–891, 2011.